Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Petra Mihalič

Učenje programiranja z okoljem Scratch

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Janez Demšar

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V slovenskem šolskem sistemu lahko pri poučevanju računalništva v prihodnjih letih pričakujemo spremembe v smeri zamenjave trenutnega načina poučevanja uporabe pisarniških programov s poučevanjem konceptov iz računalništva in, v primerni meri, tudi programiranja. Med programskimi jeziki oz. okolji, ki so primerna za poučevanje otrok, je posebej popularno okolje Scratch, ki je, v smislu pedagoškega pristopa, naslednik jezika Logo.

V diplomskem delu pripravite naloge za poučevanje programiranja z okoljem Scratch. Organizirane naj bodo po konceptih, ki jih želimo poučevati (zanke, spremenljivke, dogodki ipd.). Opišite tudi svoje izkušnje s krožka in poletne šole. Okolje Scratch primerjajte z drugimi, sorodnimi okolji.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Petra Mihalič, z vpisno številko **63110275**, sem avtorica diplomskega dela z naslovom:

Učenje programiranja z okoljem Scratch

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom prof. dr. Janeza Demšarja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 20. avgusta 2014 Podpis avtorja:

Hvala moji družini, da me je v letih študija in pred tem podpirala moralno, finančno in z iskreno ljubeznijo. Posebej gre zahvala sestri Sari, ker me je navdušila nad Scratchem in Lenartu, ki mi je pomagal pri sestavljanju Lego Mindstorms.
Hvaležna sem Kristjanu, ki me je podpiral v težjih in lepih trenutkih in verjel vame.
Hvala najboljšim prijateljem: Primožu, Primožu, Manci, Luku, Jerneji, Jaku, Ožboltu, Roku, Kristini, Petru za čudovita tri leta nesebične pomoči, sodelovanja in prijateljstva.
Zavhala gre tudi Špeli, ker je marsikatero noč slabše spala zaradi zvoka moje tipkovnice in se trudila razumeti moje računalniške zgodbe.
Najlepša hvala mentorju izr. prof. dr. Janezu Demšarju, ker me je vpeljal globoko v svet poučevanja programiranja.

Sari, ker je s Scratchem hotela pridobiti vsaj kakšno uro s sestro, a ji jo je prav Scratch največ vzel.

Kazalo

Povzetek

Za učenje programiranja otrok je na voljo kar nekaj dobrih programskih jezikov in okolij, ki omogočajo, da se začetniki spoznajo predvsem z osnovnimi programskimi konstrukti, navadno s pomočjo interaktivnega okolja. Nekaj izmed teh jezikov in okolij je opisanih v diplomski nalogi, med seboj so tudi primerjani. Scratch je jezik in programsko okolje, namenjen učenju programiranja s pomočjo blokov, ki olajšajo pisanje zapletenih ukazov in zmanjšajo začetniške težave s sintaktičnimi napakami. Zaradi pomanjkanja nalog za učenje programskih konstruktov sem s pomočjo ugotovitev iz delavnic učenja programiranja v Scratchu napisala naloge za učence druge triade osnovne šole.

Ključne besede: programiranje za otroke, Scratch, učenje programiranja.

Abstract

There are quite a few good programing languages and environments for teaching kids how to program. They help beginners learn basic programming constructs usually with help of interactive environment. In this thesis I will describe and compare a few of those languages and environments. Scratch is a language and programming environment that is also intended for learning programming using blocks that make writing complicated instructions easier and reduce beginners' difficulties with syntax errors. Due to the lack of assignments for learning programming constructs I have written a few assignments for students of second triad of primary school with help of findings from Scratch programming workshops.

Keywords: programming for kids, Scratch, learning programming.

Poglavje 1

Uvod

Danes se otroci zelo zgodaj srečajo z računalnikom. Nekatere hitro premamijo igre, spet drugi se skozi igro učijo. Veliko dobrih programerjev pravi, da so se s programiranjem začeli ukvarjati že v osnovni šoli.

Otroci oziroma mladostniki v tem obdobju še neobremenjeno razmišljajo tudi o zahtevnih programskih strukturah in elementih ter se včasih z njimi spopadajo na drugačen način kot starejši programerji in tisti, ki začnejo programirati v poznejših letih.

S pomočjo delavnic, ki sem jih pomagala voditi na Osnovni šoli Alojzija Šuštarja, sem spoznavala, kako otroci oz. najstniki dojemajo programske strukture. Na krožku, ki so ga obiskovali nadarjeni učenci 4., 5., 6. in 7. razreda devetletne osnovne šole, smo s pomočjo programskega jezika za otroke spoznavali programiranje. Poleg tega smo podobne delavnice izvedli tudi na Poletni šoli FRI: Čarajmo v Scratchu. Otroci so se s programiranjem na računalniku srečali prvič, predvsem tisti na Poletni šoli, saj so otroci s krožka leto prej že imeli tečaj po programu Računalništvo brez računalnika. Tako je učenje programiranja na nek način postalo igra in ideje za naloge iz programiranja niso prihajale le z naše strani, temveč so si zaradi prijetnega okolja tudi otroci sami izmišljali zgodbe in naloge. Program, s katerim smo spoznavali programiranje, se imenuje Scratch. Moja naloga pri tem je bila, da sem opazovala otroke, kako dojemajo posamezne elemente, zanke, strukture, objekte, dinamiko lažjih in težjih programskih objektov pri programiranju. Prednost programiranja v Scratchu je, da se otrokom ni potrebno ukvarjati s sintaktičnimi napakami, saj so bloki z ukazi že ustvarjeni, oni jih le

povlečejo v glavno okno in spreminjajo argumente. Tako se lahko še bolj posvetijo programiranju in razmišljanju o sestavi programa in ne le uporabi ukazov, s katerimi imajo programerji v začetku nemalo težav.

Na podlagi ugotovitev s projektov sem nato napisala Naloge za poučevanje krožka računalništva s pomočjo katerih sem prišla v stik tudi s sestavljalci programa za neobvezni izbirni predmet Računalništvo in tudi tam pomagala pri sestavi nalog.

V diplomi sem raziskala tudi nekaj ostalih programskih jezikov za učenje programiranja in dobila pregled nad različnimi pristopi poučevanja programiranja otrok.

Poglavje 2

Programiranje za otroke

Programiranje ni cilj sam po sebi, ampak pomaga otrokom prevzeti nadzor, rešiti probleme in zgraditi prihodnost po principu njihove predstave in kreativnosti. O pomenu poučevanja programiranja razpravlja dokument *Computer Science K–8: Building a strong foundation* [5], po katerem povzemam večino razdelka.

Logo je prvi programski jezik, ki je bil namenjen otrokom. S pomočjo Loga so (in še danes) otroci spoznavali, kako lahko uporabljajo računalnik tako, da so sami v aktivni vlogi oblikovalca in razvijalca. Otroci s tem razvijajo kompetence kritičnega razmišljanja, reševanja problemov, komunikacije, sodelovanja, kreativnosti in inovacij ... Poleg tega začnejo razvijati algoritmično in računalniško mišljenje.

Otroci poleg tega razvijejo nek koncept, s katerim se nato lotijo programiranja ali pa tudi vsakdanjih problemov v življenju. Najprej si zastavijo izziv, ki ga hočejo doseči in ga definirajo.

1. Izberejo informacije.
2. Izdelajo plan.
3. Pripravijo delujoči prototip.
4. Preizkusijo in popravijo napake.
5. Zberejo odzive ostalih.

6. Ocenijo in popravijo, kar je potrebno.

7. Objavijo in začnejo znova.

Pozitivna izkušnja v zgodnjih letih osnovnega šolanja velikokrat motivira učence, da se s programiranjem ukvarjajo tudi kasneje.

Pomemben element, ki se ga otroci naučijo skozi programiranje, je tudi refleksija. V programerskem smislu predvsem to, ali sem dosegel, kar sem želel, dobil prave rezultate v dovolj hitrem času z optimalno kodo?

Namen učenja programiranja ni, da otroke naučimo pretirane uporabe računalnika. Naš namen je, da računalnika ne uporabljajo kot orodje za dosego cilja, ampak kot ustvarjalce orodja, s katerim si bodo olajšali vsakdanje situacije. Otroci si sami sprogramirajo program, ki jim bo olajšal delo.

Razlogi, zakaj je dobro otroke učiti programiranja:

1. Razmišljanje je dobro za razmišljanje.

2. Sestavljanje nove generacije kreatorjev in inovatorjev.

3. Spodbujanje učencev, da spremenijo svet.

4. Sodelovanje, komunikacija, timsko delo.

Učenje programiranja pomaga pri vztrajnosti. Otroci namreč programirajo in popravljajo program, dokler jim ne uspe izdelati delujoče kode. Poleg tega se učijo dekompozicije, saj so pri programiranju obsežnejših programov primorani probleme razdeliti na krajše in enostavnejše probleme. To znanje jim lahko koristi tudi pri drugih predmetih in v življenju na sploh, kar pomeni, da z učenjem programiranja pridobivajo znanje tudi za širše področje.

2.1 Programiranje na slovenskih osnovnih šolah

Z novim šolskim letom 2014/15 se v slovenske osnovne šole vpeljuje neobvezni izbirni predmet Računalništvo od 4. do 6. razreda. Že prej je obstajal izbirni predmet Računalništvo za učence tretje triade, vendar njegov poudarek ni bil na

spoznavanju s programiranjem in programskimi konstrukti, ampak bolj na urejanju besedil, spoznavanju računalniških omrežij in multimedije.

Program za novi izbirni predmet Računalništvo je podoben, kot je predvideno izvajanje krožka računalništva, za katerega sem sestavila naloge, ki se nahajajo v prilogi. Te naloge so uporabljene tudi v predlaganem učnem načrtu za izvajanje neobveznega izbirnega predmeta Računalništvo Zavoda Republike Slovenije za šolstvo. Tudi na osnovnih šolah bo poudarek na spoznavanju s programskimi koncepti in njihovo dejansko uporabo. Prav tako so na spletni učilnici uporabljene razlage konceptov iz Nalog za krožek računalništva.

Poleg programiranja v Scratchu je želja, da se otroci na krožku naučijo računalniškega razmišljanja in pridobijo veščine za reševanje problemov. Pri tem krožku oziroma neobveznem izbirnem predmetu se poučuje le programiranje in ne ostalih področij računalništva in njegove uporabe [8, 9].

2.1.1 Starost otrok

Ob sestavi diplomske naloge sem sprva razmišljala o učenju programiranja za prvo triado. Lahko bi se učili s pomočjo še enega Scratchevega izdelka: Scratch junior, kjer ni potrebno biti pismen, da lahko programiraš, vendar za enkrat še ni dostopen vsem, saj je ta trenutek razvit le za uporabnike tabličnih izdelkov znamke Apple. To bi bil za osnovne šole prevelik finančni zalogaj. V prvi triadi bi se prav tako lahko spoznali z osnovami programa Vidra.

Omejila sem na drugo triado osnovnošolcev. Ti že znajo brati in poznajo osnovne matematične ukaze ter jim logično razmišljanje in sklepanje ne delata večjih preglavic. Otroci so tudi v tem obdobju najbolj dovzetni za ustvarjanje preprostih in zanimivih iger, ki jih nato lahko tudi igrajo.

V zadnji triadi osnovne šole je že ponujen izbirni predmet Računalništvo, ki sicer pokriva drugo področje, vendar je v učnem načrtu tudi nekaj učenja o algoritmih. V tretji triadi bi se programiranja lahko lotili že s pravimi programskimi

jeziki, kjer ne bi bilo potrebno toliko grafičnih elementov, da bi mladostnike pritegnili k uporabi [8, 9].

Poglavje 3

Programski jeziki

V nadaljevanju bom opisala nekaj programskih jezikov in okolij za otroke ter jih primerjala med seboj. Še posebej bom dala poudarek primerjavi z izbranim programskim jezikom Scratch, ki se mi je od vseh jezikov zdel še najbolj dovršen in primeren za učenje programiranja otrok. Na koncu vsakega opisa in primerjanja bom podala še primer naloge, ki bi jo lahko v tem jeziku in okolju reševali učenci v osnovni šoli.

3.1 Logo

Logo[1] je prvi programski jezik, ki je bil napisan specifično za otroke. V letu 1967 so ga ustvarili Daniel G. Bobrow, Wally Feurzeig, Seymour Papert in Cyntia Solomon. Osnovan je bil na že obstoječem programskem jeziku LISP. Danes poznamo brezplačne programe za računalnike, kot je na primer za uporabnike Oken MSWLogo ali Mac uporabnike ACSLogo [10].

Logo je programski jezik, ki je narejen za učenje programiranja, predvsem matematičnih konstruktov s pomočjo želvjega kurzorja. Samo okolje sicer ni pretirano barvito in zato ne tako atraktivno za otroke kot nekateri drugi programski jeziki za učenje, vendar je preprost ter uporaben. Ta programski jezik je predvsem zaželen zaradi svoje uporabnosti pri učenju grafike, kotov ter koordinatnega sistema.

Osnovni ukazi pri Logu so ukazi za risanje: naprej, nazaj, levo, desno, brisanje

[1] `http://el.media.mit.edu/logo-foundation/index.html`

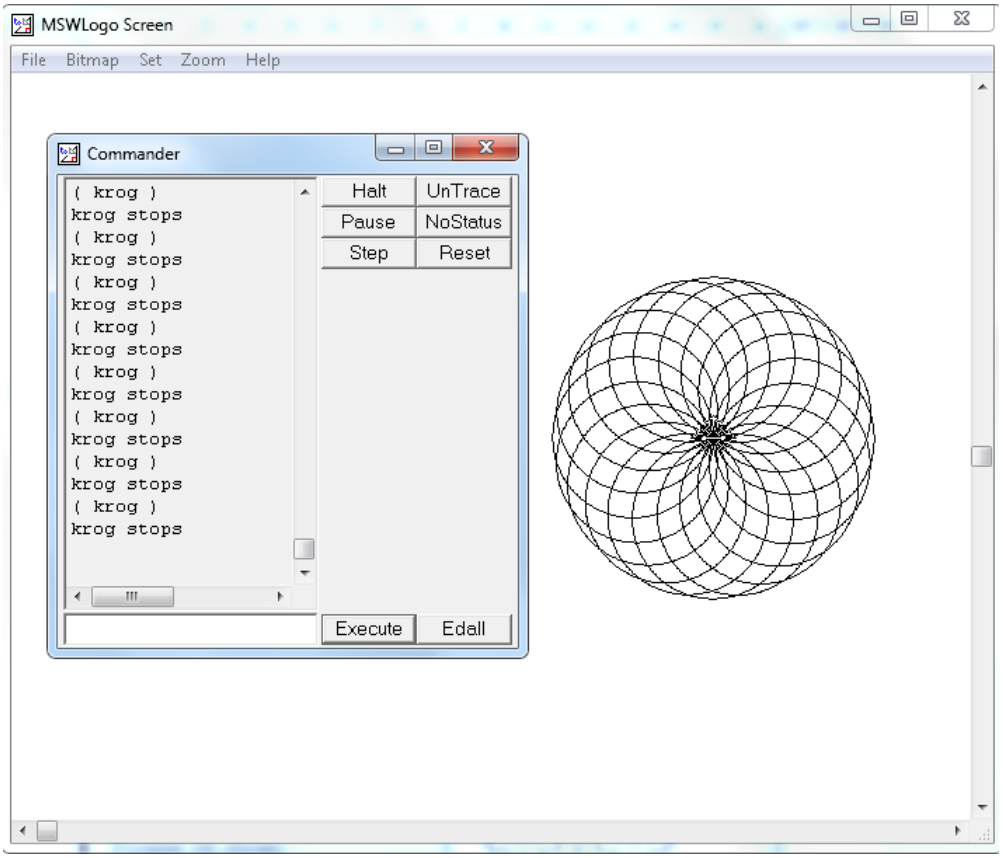
zaslona (forward, backward, right, left, clearscreen). Za temi ukazi stojijo argumenti, ki pomenijo enote za premikanje ali kote. Pri Logu se po površini premikamo po koordinatah. Obstajajo tudi ukazi za dvig, spust svinčnika, prikaz želve, postavljanje koordinat, oznak. Ima tudi spremenljivke, ki se uporabljajo tako, da ob deklariranju podamo ime z narekovajem ("spremenljivka), vrednost pa dobimo tako, da napišemo :spremenljivka. Tipa spremenljivke nam ni potrebno vnaprej določiti. Aritmetične operacije, ki jih poznamo v Logu, so: seštevanje, odštevanje, množenje, deljenje, korenjenje ... Ukaz random zgenerira naključno vrednost med 0 in zapisanim argumentom. Logo podpira tudi funkcije in rekurzijo.

Logo je funkcijski programski jezik.

Primerjava s programskim jezikom Scratch:
Jezik je bolj usmerjen v programiranje kot Scratch, saj se pri programiranju nalog v tem jeziku ne ukvarja toliko z izgledom samih figur in okolja, ampak se osredotoči na samo pisanje ukazov, premikanje elementov po prikaznem polju ter funkcionalnost programa. To je prav tako kritika, saj se lahko določeni otroci hitro naveličajo enoličnosti izgleda površine.

Pri Logu se ukaze piše samostojno, brez uporabljanja blokov ali prej pripravljenih ikon, tako da se otrok uči tudi pisanja pravilne in natančne sintakse. Pri tem ima tako več težav s sintaktičnimi napakami kot pa učenec, ki programira v Scratchu. Slednji uporablja bloke z napisanimi ukazi po sistemu 'povleci in spusti'. Pri Scratchu imajo mladi programerji dostop do ogromne knjižnice izdelkov svojih sovrstnikov in drugih ustvarjalcev ter tako dobijo še več idej, ki se jih sami morda ne bi spomnili. Pri Logu otroci nimajo možnosti, da bi svoje projekte delili z drugimi in imeli vpogled v izdelke drugih. [25]

Primer naloge in grafični izgled spletne aplikacije:
Primer se nahaja na sliki 3.1.
Nariši krog in shrani ukaz v funkcijo: to krog
repeat 360 [forward 1 left 1]
end
Nato večkrat nariši krog s pomočjo funkcije, tako da začenjaš pod različnimi koti in tako narišeš rožo:
repeat 20 [krog rt 18]



Slika 3.1: Izgled aplikacije v Logu

3.2 Lego Mindstorms

Lego Mindstorms[2] je komplet programske in strojne opreme, kjer s pomočjo sestavljanja Lego kock ustvarjamo robote in ostale naprave ter nato zanje napišemo program, da se naprava premika tako, kot sami želimo. Razvit je bil leta 1994, kot izobraževalno orodje se prodaja v partnerstvu s podjetjem Lego in MIT Media Laboratory. Poleg učenja programiranja ta program pomaga pri razvijanju občutka za konstruiranje različnih objektov, ki jih lahko sestavimo s pomočjo Lego kock [17].

Najnovejši set robotov z imenom Lego Mindstorms EV3 vsebuje motorje, senzorje, programabilne bloke, daljinski nadzor ... Postopek za izdelavo programa je

[2] `http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com`

preprost [7]:

1. razmisli, kaj želiš ustvariti,
2. zgradi robota iz Lego kock in vključi potrebne senzorje, motorje ali kaj drugega, kar potrebuješ za izvedbo zamišljenega programa,
3. sprogramiraj program v programskem vmesnniku Lego Mindstorms,
4. usmerjaj robota s pomočjo pametne naprave in aplikacije Lego Mindstorms ali direktno preko programabilnih blokov.

Primerjava s programskim jezikom Scratch:
Z Lego Mindstorms se lahko igrajo tudi mlajši otroci kot pri Scratchu. Razlika med njima je namreč v tem, da so Lego Mindstorms bloki ikone, večinoma brez besedila in jih lahko razumejo tudi predšolski otroci, čeprav je za njih primernejši sklop Lego WeDO[3].

Še ena izmed opaznejših razlik je, da lahko pri Lego Mindstorms otroci v realnem prostoru opazujejo sprogramiran program in ga fizično preizkušajo, medtem ko je pri Scratchu izvajanje programa omejeno na grafično površino, namenjeno prikazu programa.

Glede na Scratch je tu možne veliko več interakcije preko vhodov, izhodov, odzivanja na fizične dogodke in sprejemanja podatkov, ki jih lahko robot zazna iz okolice. Vendar ima tudi Scratch možnost razširitev s komponentama PicoBoard in LEGO WeDo, kjer lahko prav tako programiramo programe za drugo strojno opremo, ki ni le grafična površina.

Ena izmed kritik Lego Mindstorms bi lahko bila, da je strojno opremo potrebno kupiti, česar pa si ne more privoščiti vsaka šola ali gospodinjstvo.

Tudi tu je, kot na Scratchevi spletni strani, mogoče objavljanje programov, slik in vprašanj na spletni strani[4], kjer lahko otroci dobijo ideje za svoje programe.

Primer naloge in grafični izgled spletne aplikacije:
Sestavi robota, ki se bo premikal naravnost in imel senzor za dotik ter barvni senzor.

[3] `http://education.lego.com/es-es/preschool-and-school/lower-primary/7plus-education-wedo`

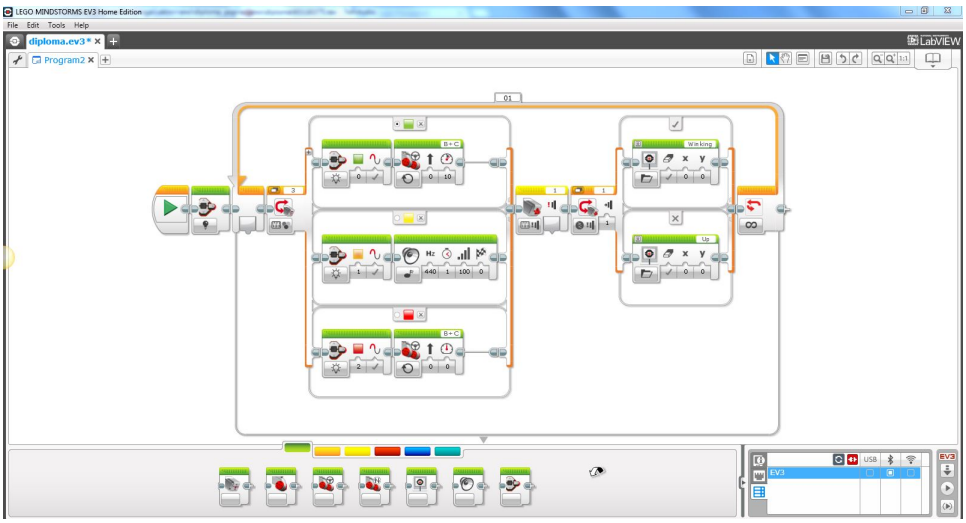
[4] `http://www.us.lego.com/en-us/mindstorms/community`

Napiši mu program, ki v začetku ugasne luč med smernimi tipkami na robotu. Če senzor zazna zeleno luč, se robot premika naravnost, luč med smernimi tipkami pa naj utripa zeleno. Če senzor zazna rumeno barvo, naj robot zapiska in luč naj spremeni barvo v rumeno. V primeru, ko zazna rdečo, naj se ustavi, luč med tipkami, naj zasveti rdeče.

V času, ko robot ne zazna nobene od naštetih barv, naj se izvaja tako, kot pri zaznani zeleni barvi.

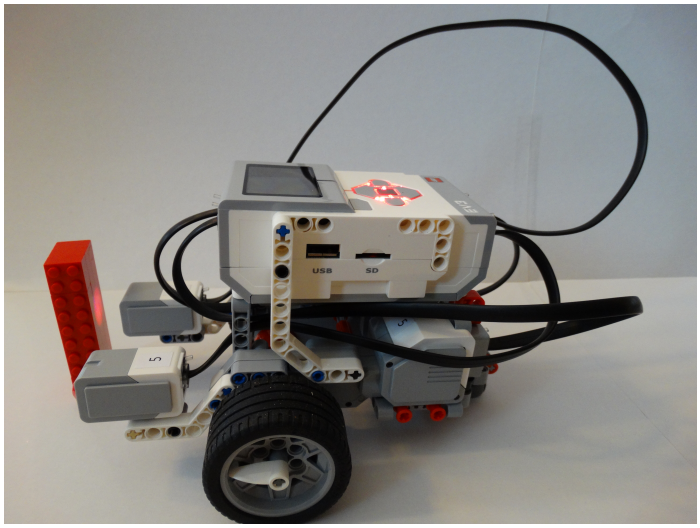
Če med izvajanjem programa pritisnemo rdečo tipko za zaznavanje dotika, naj robot pomežikne. Če tipka ni pritisnjena, naj bodo na zaslonu izrisane odprte oči.

Za lažje razumevanje navodil, si lahko demonstracijo naloge pogledate na povezavi[5]. Slikovni material za nalogo se nahaja na slikah 3.2 in 3.3.



Slika 3.2: Izgled programske kode v programu Lego Mindstorms

[5] https://www.youtube.com/watch?v=3Lfq9kzToq8&feature=youtu.be



Slika 3.3: Sestavljen robot, primeren za pripravljeno nalogo

3.3 Greenfoot

Greenfoot[6] je interaktiven programski jezik, ki je bil sprva namenjen srednješolcem kot pripomoček pri učenju programiranja, in sicer od leta 2003. Razvili so ga na Univerzi Kent, danes ga sponzorirata tudi Oracle and Google [1]. Ta programski jezik temelji na dvodimenzionalnem grafičnem prikazu elementov, ki jih usmerjamo s pomočjo pisanja programske kode. Programiranje v Greenfootu je objektno.

Interaktivno razvojno okolje, ki temelji na programskem jeziku Java, je bilo razvito v izobraževalne namene. Omogoča enostaven razvoj dvodimenzionalnih grafičnih aplikacij, simulacijo in razvoj interaktivnih iger. Okolje vključuje urejanje projekta, avtomatsko dopolnjevanje, označevanje sintaktičnih napak, s čimer so uporabniki direktno usmerjeni k učenju programiranja v zahtevnejših jezikih, v tem primeru v Javi. Na voljo je na operacijskih sistemih Windows, Mac OS X, Linux ... Programski model Greenfoota sestavlja razred World, ki ga predstavlja pravokotna površina, na kateri se odvija vse, kar uporabniki sprogramirajo. Poleg razreda World lahko uporabljamo še actor – objekte, ki so predstavljeni na prej omenjeni površini. Te objekte lahko programiramo neodvisno med seboj, na površini pa jih je lahko neomejeno število. Svet in objekti so predstavljeni kot javanski objekti

[6] `http://www.greenfoot.org/`

ter definirani z javanskimi razredi.

Objekte lahko tako brez težav premikamo, rotiramo, spremenimo izgled, detektiramo ... V Greenfootu programiramo s podrazredi. Greenfoot omogoča tudi API-metode za animacijo, zvok, naključna števila in manipuliranje s slikami. Uporabljajo se lahko standardne javanske knjižnice.

Greenfoot je narejen vizualno prijetno, tako da učencem približa lahek dostop do animiranih grafičnih podob, zvoka in interakcije. Okolje je izredno interaktivno in spodbija raziskovanje. Poleg tega je sestavljeno tako, da očitno ilustrira pomembne koncepte objektno orientiranega programiranja, kot so relacije med razredi in objekti, metode, parametri in interakcije med objekti. Cilj razvijalcev Greenfoota je, da uporabnikom kar najbolj pravilno predstavijo objektno programiranje [19].

Primerjava s programskim jezikom Scratch:
Greenfoot je v nasprotju s Scratchom veliko zahtevnejši, saj tu ni blokov, ki bi jih potegnili in spustili ter s tem sestavili kodo. Vendar je še vedno zelo dobro poskrbljeno, da mladi programerji nimajo težav s sintakso, saj ob zaganjanju programa Grenfoot lepo označi, kje se nahajajo napake v kodi, ob pisanju kode pa avtomatsko poda predloge za dokončanje ukaza. Okolje ima lep uporabniški vmesnik, ki je pregleden in se dobro odziva na dogodke. Nekaj objektov je v prvem projektu že avtomatsko dodanih, vendar jih uporabnik lahko doda ali spremeni, kar mu omogoča razvoj lastnih aplikacij in idej. Pri programiranju je omogočena uporaba že napisanih metod za posamezne objekte.

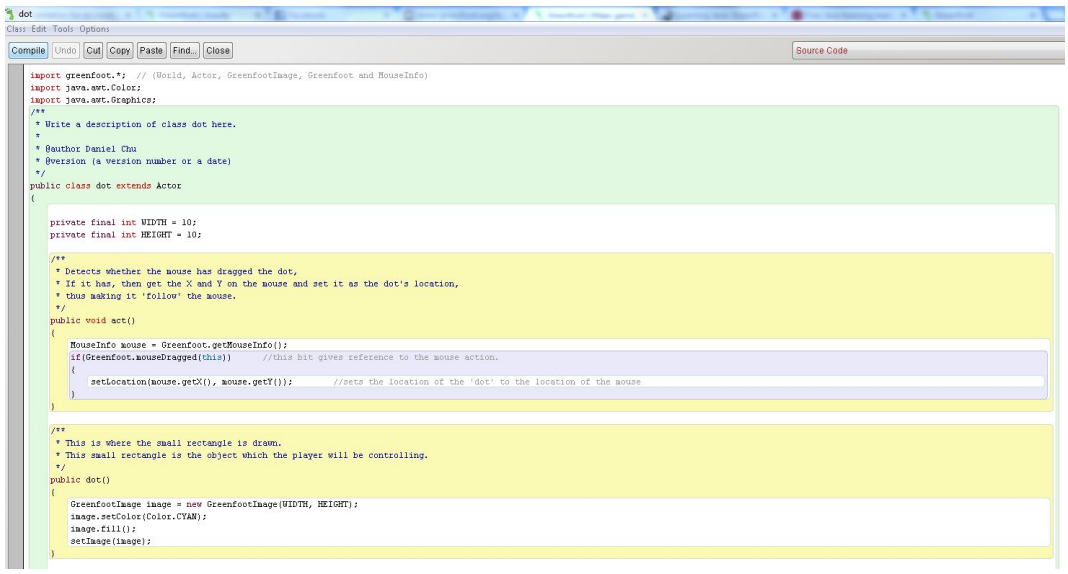
Prav tako kot Scratch ima tudi Greenfoot možnost deljenja ustvarjenih iger na spletni strani[7] in omogočeno komentiranje. Nalaganje projekta na spletno stran ni zahtevno, saj je vedno v desnem zgornjem kotu prisoten gumb za deljenje projekta, kjer lahko uporabnik doda še informacije o njem.

Greenfoot je nekoliko zahtevnejši kot Scratch in je primeren za starejšo populacijo otrok. Z Greenfootom je prehod na prave programske jezike lažji kot v Scratchu, še posebej na Javo, ki je osnova Greenfoota [22, 19].

Primer naloge in grafični izgled aplikacije:
Kot primer igre lahko sprogramiramo labirint z več nivoji. Piko nato z miško povlečemo po belem polju do cilja. Če se vmes dotakne sten, se igra konča. Če

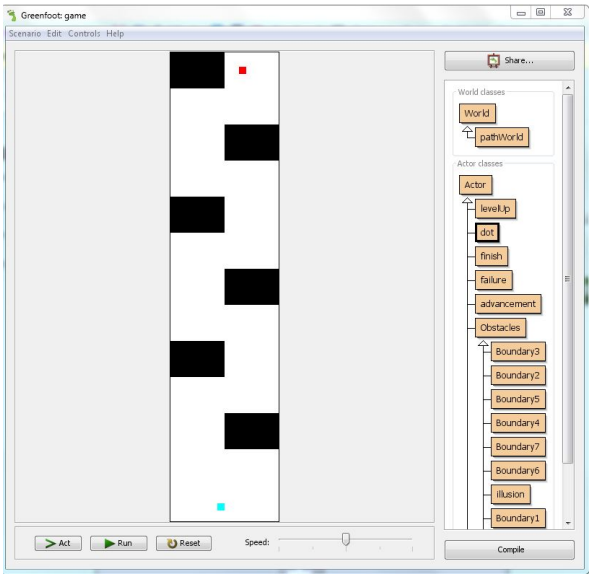
[7] `http://www.greenfoot.org/`

pridemo do cilja brez dotika sten, se izriše naslednji nivo.[8]
Primer grafičnega izgleda aplikacije in programske kode se nahaja na slikah 3.4 in 3.5.



Slika 3.4: Izgled programske kode v Greenfootu

[8]Vir naloge Maze game je dostopen na: `http://www.greenfoot.org/scenarios/726`



Slika 3.5: Izgled aplikacije v Greenfootu

3.4 Alice

Alice[9] je programsko okolje za 3D programiranje, ki omogoča enostavno animacijo za pripovedovanje zgodb, igranje in sestavljanje interaktivnih iger ali video posnetkov. Programer jih na spletu lahko tudi deli. Nastal je leta 1998 na University of Virgina in Carniege Mellon University. Programski jezik temelji na objektno usmerjenem, dogodkovno vodenem programiranju. Po taksonomiji je podoben jeziku Standard ML. Alice omogoča uporabo in kreiranje metod, funkcij, spremenljivk, parametrov in rekurzije. Je objektno usmerjen programski jezik, namenjen učenju programiranja z IDE. Vendar je bil program razvit z namenom izobraževanja, zato ne uporablja kompleksne semantike, ki jo najdemo v pravih programerskih jezikih [2, 16, 4].

Tudi tu se programira po načelu 'potegni in spusti'. Z uporabo blokov je programer tako neobremenjen in ni potrebno, da si zapomni sintaktične ukaze. Kreativnost se spodbuja tudi v tem okolju, saj lahko uporabnik sestavi program po lastni ideji in ga pri tem okolje tudi spodbuja, saj lahko sam dodaja objekte in jih ureja. Z že podanimi 'potegni in spusti' elementi se spodbuja otroško razisko-

[9] `http://www.alice.org/index.php`

vanje. Ker so bloki že podani, jih otrok želi preizkusiti in se na ta način spozna s programskimi konstrukti, ki jih drugače mogoče ne bi uporabil, ker se ne bi zavedal, da sploh obstajajo.

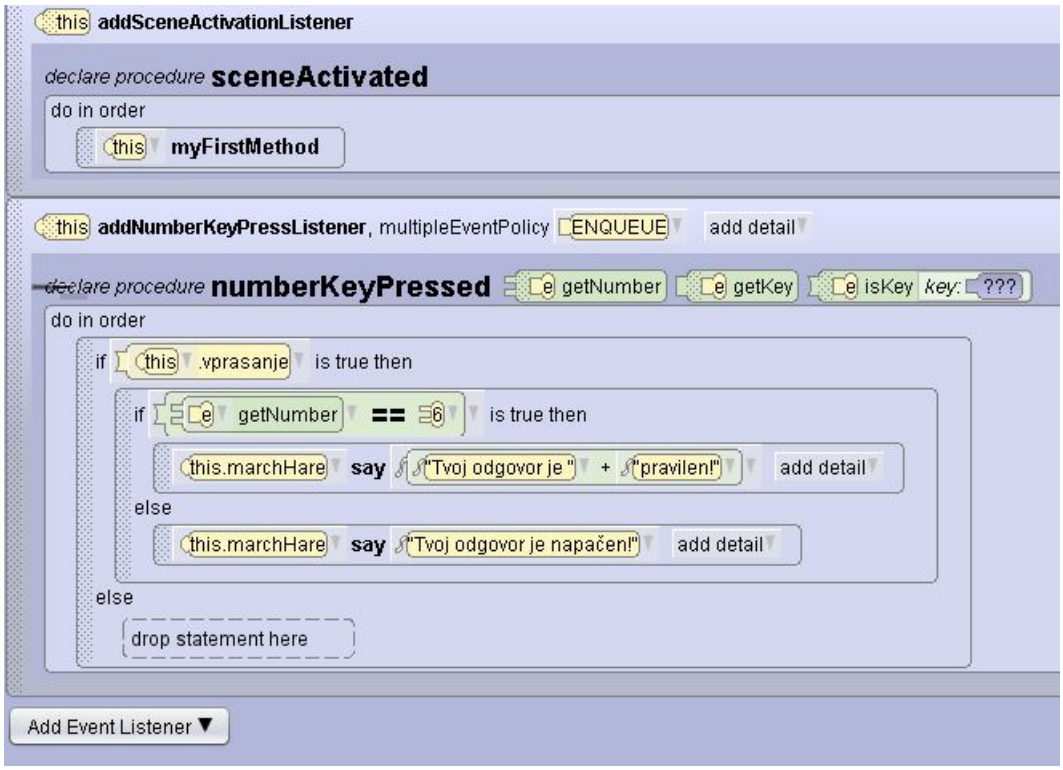
Alice je tako kot Scratch primerna tudi za pripovedovanje zgodb, kar pa razvija najbolj osnoven programski konstrukt: zaporedje izvajanja ukazov. Eden izmed ciljev razvoja programskega okolja in jezika Alice je bil tudi, da privabijo več deklet k programiranju, saj ponuja veliko možnosti za oblikovanje in ustvarjanje [18, 28]. Preden otroci začnejo s programiranjem, je dobro, da si odgovorijo na vprašanja in s tem načrtujejo svoje delo. Za to obstajajo celo vprašanja, ki so jih pripravili razvijalci in jih objavili na spletni strani [3]:

Najprej napiši opis ali scenarij svoje zgodbe ali programa, ki ga želiš predstaviti. Dolg naj bi bil 1–2 odstavka. Razmisli o:

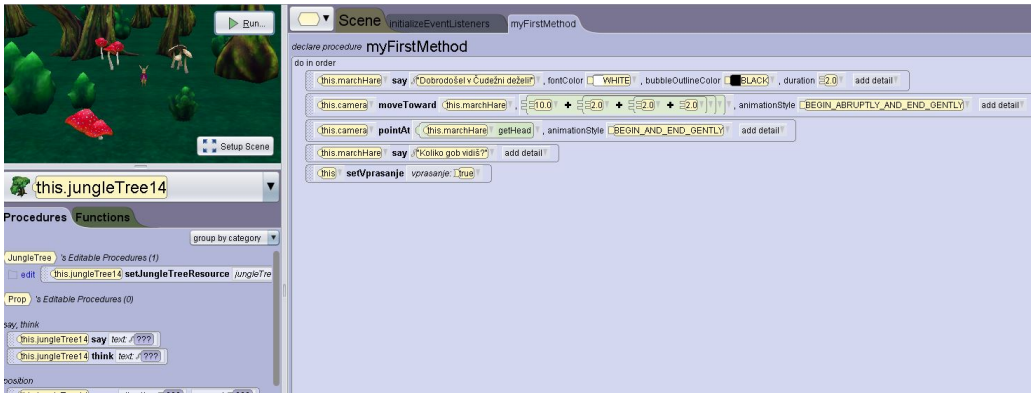
1. Katero zgodbo želiš povedati?
2. Katere objekte ali avatarje potrebuješ?
3. Kateri objekt bo igral glavno vlogo v zgodbi (medtem ko bodo ostali objekti uporabljeni za ozadje scenarija)?
4. Katere dogodki se bodo odvijali v zgodbi (Dogodki/akcije v zgodbi bodo najverjetneje postali ukazi v programu)?

Primerjava s programskim jezikom Scratch:
Alice je precej podobna Scratchu glede pogleda in grafične površine, na kateri prikažemo rezultat programiranja. Razlikuje se v tem, da je grafična površina v 3D obliki. Bloki so sicer podobni, saj so tudi tu na način 'povleci in spusti', vendar so ukazi zahtevnejši. Spodbujanje kreativnosti, ustvarjalnosti in pripovedovanja zgodb pa se spodbuja na enak način kot pri Scratchu. Dovoljena je namreč uporaba domišljije in velika svoboda pri izbiri ozadja, likov in premikanja le-teh. Vendar je Alice zahtevnejša od Scratcha. Skupaj z Greenfootom sta primerna za starejše uporabnike, kot je Scratch, in sta zelo podobna oziroma poenostavljena javanska programska jezika. Pri Alice je zahtevnejše tudi ustvarjanje in urejanje 3D elementov, ki zahtevajo drugačno obdelavo in več le-te kot 2D elementi pri Scratchu [22].

Primer naloge in grafični izgled spletne aplikacije:
Naloga, ki sem jo kot primer sprogramirala v Alice, je, da nas zajec najprej pozdravi v Čudežni deželi. Obdan je z gozdom, ki je podoben tistemu v zgodbi Alica v Čudežni deželi. Kamera se zajcu približa in ta vpraša, koliko gob vidi uporabnik. Ko uporabnik odgovori, mu zajec pove, ali je pravilno odgovoril. Če je odgovoril napačno, mu sporoči, da je bil odgovor nepravilen.
Primer rešitve kode se nahaja na slikah 3.6 in 3.7.



Slika 3.6: Izgled programske kode v Alice



Slika 3.7: Izgled aplikacije in programske kode v Alice

3.5 Kodu

Kodu[10] je vizualni program za učenje programiranja, predvsem za ustvarjanje iger, ki je na trgu od leta 2009. Razvit je bil predvsem za uporabnike Microsoftovih izdelkov.

Tudi tu ne tipkamo kode, ampak uporabljamo vizualne elemente preko krmilnika igre.

Prikaz v grafični konzoli je v 3D obliki, kar je podobno kot pri programskem jeziku Alice.

Programiranje v Koduju je poenostavljeno in tudi tu ni tako kompleksno, kot je v naprednejših programskih jezikih. Lahko programiramo preko igralnega krmilnika ali v kombinaciji s tipkovnico in miško. Lahko uporabljamo simbolične spremenljivke, vejitve, zanke, manipulacijo z nizi in števili, podprograme, polimorfizem ... Preprostost programiranja je dosežena z dobro simuliranim okoljem, saj je v ozadju veliko več programske kode, kot jo vidi uporabnik. Programiranje v Koduju razvija otrokovo prostorsko predstavo, pomaga pri učenju geometrijskega modeliranja in grafike. Je vizualno programsko okolje, ki ne uporablja blokov z napisanimi ukazi, ampak so ukazi v obliki ikon. Ikone so v obliki puzel, ki jih

[10] `http://www.kodugamelab.com/`

sestavljamo in jim dodajamo različne pogoje ali ukaze. Zaporedno lahko sestavimo samo ukaze, ki si sintaktično lahko sledijo. S tem je onemogočeno napačno programiranje zaporedja ukazov, ki zagotovo ne sodijo skupaj.

Programiramo lahko kar medtem, ko smo v igri postavljeni v prostor. Tam lahko dodajamo elemente, jim pišemo kodo, jih urejamo. Vse to se dogaja v interaktivnem 3D svetu. S preprostostjo in dobrimi vizualnimi učinki uporabniku olajša programiranje in iskanje ukazov. Ti so razdeljeni na stavka when in do. Pri delu when imamo ponujenih veliko možnosti, kateri dogodek bo sprožil pogojni stavek. Pri delu do pa objektu, ki ga programiramo, podamo, kako naj se odzove. Tudi tu je na ikonah grafično narisanih ogromno možnosti. [12]

Primerjanje s programskim jezikom Scratch:
V primerjavi s Scratchem je Kodu drugačen v dimenzijah, saj je v 3D grafiki. Ima tudi narisane ikone in na blokih ni napisano besedilo, kot je to pri Scratchu. Tako se z njim teoretično lahko igrajo tudi manjši otroci, vendar je za njih to nekoliko prezahteven svet ravno zaradi 3D grafike. Čeprav je jezik pisan v ikonah, se še vedno pojavljajo angleška navodila ob vstopu v igro in navodilih za dodatne možnost ob robu ekrana.

Poudarek pri programiranju je predvsem na pogojnih stavkih in odzivanju na dogodke. Veliko manj ima možnih blokov in ponujenih programskih konstruktov, kot je to možno v Scratchu. Ni namreč mogoče uporabljati tabel in definiranja svojih funkcij.

Namenjen je izključno razvijanju iger in ne animaciji ali le pripovedovanju zgodb.

Primer naloge in grafični izgled spletne aplikacije:
Naloga naj bo, da Kodu tava po svetu in nabira školje. Ko se školje s klikom miške dotakne, naj se mu prištejejo točke, školjka pa naj izgine.

Primer rešitve programa se nahaja na slikah 3.8, 3.9, 3.10 in 3.11.

Slika 3.8: Izgled programske kode v programskem jeziku Kodu za Koduja

Slika 3.9: Izgled aplikacije in programske kode v programskem jeziku Kodu za školjko

Slika 3.10: Primer izbiranja blokov z ikonami v programskem jeziku Kodu

Slika 3.11: Izgled grafične površine v programskem jeziku Kodu

3.6 AppInventor

AppInventor[11] je bil razvit v letu 2009, javno dostopen je od leta 2010, ko se je trg aplikacij za operacijski sistem Android že dobro razširil. Veliko uporabnikov je začutilo potrebo po razvijanju aplikacij, ki jim bi olajšale uporabno mobilnih telefonov in življenja na splošno. Googlovi inženirji so zaradi velikega zanimanja razvoj aplikacij omogočili tudi začetnikom programiranja oziroma otrokom in pripravili aplikacijo, namenjeno predvsem starostni skupini od 10 do 18 let in več. AppInventor je spletna aplikacija, s katero kreiramo aplikacije za mobilne telefone. Poleg spletnega vmesnika lahko aplikacijo preizkušamo na simulatorju ali pa jo direktno preizkusimo kar na telefonu, ki je povezan v internetno omrežje, ima aplikacijo AppInventor in kodo za dostop do aplikacije.

Prednost aplikacije je v tem, da veliko otrok in najstnikov redno uporablja svoje mobilne naprave ali tablice. To pomeni, da ob njih preživijo veliko časa, vmes pa včasih začutijo tudi potrebo, da bi kakšen program izboljšali oziroma ga sami izdelali in so tako stimulirani, da razvijejo svoj program [30].

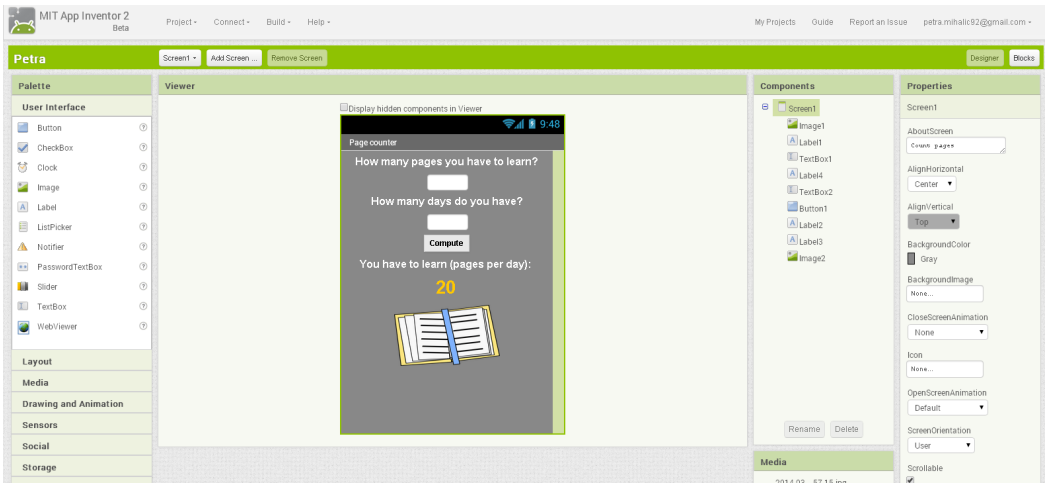
Primerjava s programskim jezikom Scratch:
Razvoj aplikacij za mobilne telefone AppInventor ima nekaj posebnosti glede na ostala otroška okolja za programiranje. Edini ima simulator za preizkušanje spisanega programa na telefonu.

Preostale funkcionalnosti in izgled so podobni ostalim programskim jezikom za otroke, saj tudi ta vsebuje 'povleci in spusti' bloke, ki omogočajo manj sintaktičnih napak pri razvoju. Programiranje aplikacije izgleda tako, da se mladi razvijalci najprej posvetijo grafični podobi, ki vsebuje standardne gumbe, vnosna polja in ostale gradnike, potrebne za uporabniški vmesnik, ter še precej ostalih podpor za postavitev strani, ostale medije, risanje, animacijo, senzorje, povezovanje ... Nato pa lahko s pomočjo blokov sprogramirajo funkcionalnost prej dodanim komponentam. Tudi tu imamo vnaprej pripravljene bloke v obliki puzel, ki jim lahko spreminjamo, na kateri gradnik se nanašajo, na kateri dogodek se odzovejo in kako nanj odreagirajo. Klicanje teh razredov in njihovih metod je zelo podobno kot v kar nekaj višjih programskih jezikih, kar pomeni, da je otrokom, ki programirajo z AppInventorjem precej lažje preiti na kakšen drug programski jezik, kot so npr.:

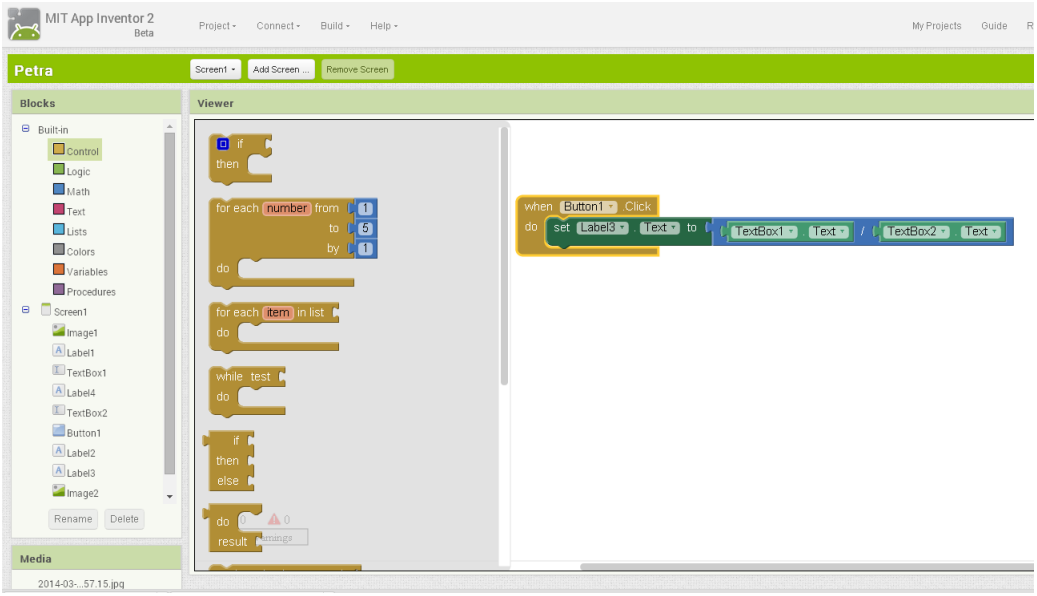
[11] `http://appinventor.mit.edu/explore/`

Java, C, C#.

Primer naloge in grafični izgled spletne aplikacije:
Aplikacija naj ima prijazen, a enostaven grafični vmesnik. V prvo prikazno okno naj uporabnik s številko zapiše, koliko strani se mora uporabnik še naučiti, v drugo vnosno polje pa, koliko dni mu za to še preostane. Ko klikne na gumb 'Compute', se mu izpiše, koliko strani na dan se mora naučiti. Sam programerski del je zelo nezahteven in kratek, vendar je pri otrocih zanj potrebno nekaj razmišljanja. Izgled aplikacije in programske kode se nahaja na slikah 3.12 in 3.13.



Slika 3.12: Izgled aplikacije v AppInventorju



Slika 3.13: Izgled programske kode v AppInventorju

3.7 Python

Python je programski jezik, ki je bil ustvarjen leta 1990. Ima dinamične podatkovne tipe, samodejno upravlja s pomnilnikom ter podpira naslednje paradigme: funkcijsko, proceduralno, strukturirano in objektno. Python ni bil razvit kot učno okolje za otroke, zato tudi ni grafično tako dodelan in poenostavljen kot preostali jeziki, ki sem jih analizirala. Pa vendar se lahko uporablja za učenje programiranja otrok, ker je hitro in lepo berljiv. Zasnovan je tako, da ima urejeno vizualno predstavitev. Ključne besede so v angleškem jeziku. Python ima manj sintaktičnih izjem in posebnih primerov kot npr.: C ali Pascal [14].

Problem, ki bil lahko bil moteč za otroke, vendar se lahko iz tega naučijo tudi lepega programiranja, je ta, da je potrebno biti zelo natančen pri uporabi presledkov in tabulatorjev. Vendar je zaradi tega koda pregledna in ni potrebno uporabljati raznih oklepajev in drugih znakov za omejevanje blokov.

Pythonova koda je v primerjavi z ostalimi programskimi jeziki lažje berljiva.

Priporočen je za učenje programiranja tudi starejšim začetnikom, saj je pisanje v Pythonu preprosto, ukazi so kratki in ni potrebno uporabljati neobičajnih simbolov, kot so: {, }, # in $ [20].

Filozofija razvijalcev Pythona je bila: Lepše je boljše kot grdo. Eksplicitno je boljše kot implicitno. Enostavno je boljše kot kompleksno. Kompleksno je boljše kot zapleteno. Berljivost šteje [23].

Primerjava s programskim jezikom Scratch:
Python za razliko od Scratcha ni bil razvit kot programski jezik, namenjen za učenje otrok programiranja. Zato tudi njegova programska okolja nimajo tako privlačne in barvite grafične podobe, kot jo ima Scratch. Prav tako v okoljih za programiranje v Pythonu ni že sestavljenih blokov, ki bi jih lahko samo 'povlekli in izpustili', da bi sestavili program. Je pa zato pravi programski jezik, v katerem se sicer hitro lahko zatipkamo, pozabimo kakšen presledek, a se prav s tem učimo prave oblike in urejenosti programa ter natančnega tipkanja in podajanja ukazov. In prav te veščine bodo tudi otroci, ko bodo iz Scratcha napredovali v resnejši programski jezik, potrebovali.

Primer naloge in grafični izgled spletne aplikacije:
Program naj ima na grafični površini izrisano ploščico, ki jo je s pomočjo tipkovnice možno usmerjati levo in desno. Ta naj lovi odbijajočo se žogico. Če žogica pade na tla, se igra konča.[12].

Vizualni izgled enega izmed Pythonovih programksih okolij je predstavljen na sliki 3.14.

```
from Tkinter import *
import random
import time

class Ball:
    def __init__(self, canvas, paddle, color):
        self.canvas = canvas
```

[12]Vir programske kode: Jason R. Briggs. Python for Kids. Dostopno na: `http://it-ebooks.info/read/2226/,2013.`

Slika 3.14: Izgled aplikacije v Pythonu

```
        self.paddle = paddle
        self.id = canvas.create_oval(10, 10, 25, 25,
                fill=color)
        self.canvas.move(self.id, 245, 10)
        starts = [-3, -2, -1, 1, 2, 3]
        random.shuffle(starts)
        self.x = starts[0]
        self.y = -3
        self.canvas_height = self.canvas.winfo_height()
        self.canvas_width = self.canvas.winfo_width()
        self.hit_bottom = False

    def hit_paddle(self, pos):
        paddle_pos = self.canvas.coords(self.paddle.id)
        if pos[2] >= paddle_pos[0] and pos[0]
                <= paddle_pos[2]:
            if pos[3] >= paddle_pos[1] and pos[3]
                    <= paddle_pos[3]:
                return True
```

```
        return False

    def draw(self):
        self.canvas.move(self.id, self.x, self.y)
        pos = self.canvas.coords(self.id)
        if pos[1] <= 0:
            self.y = 3
        if pos[3] >= self.canvas_height:
            self.hit_bottom = True
        if self.hit_paddle(pos) == True:
            self.y = -3
        if pos[0] <= 0:
            self.x = 3
        if pos[2] >= self.canvas_width:
            self.x = -3

class Paddle:
    def __init__(self, canvas, color):
        self.canvas = canvas
        self.id = canvas.create_rectangle(0, 0, 100,
                10, fill=color)
        self.canvas.move(self.id, 200, 300)
        self.x = 0
        self.canvas_width = self.canvas.winfo_width()
        self.canvas.bind_all('<KeyPress-Left>',
                self.turn_left)
        self.canvas.bind_all('<KeyPress-Right>',
                self.turn_right)

    def draw(self):
        self.canvas.move(self.id, self.x, 0)
        pos = self.canvas.coords(self.id)
        if pos[0] <= 0:
```

```
            self.x = 0
        elif pos[2] >= self.canvas_width:
            self.x = 0

    def turn_left(self, evt):
        self.x = -2

    def turn_right(self, evt):
        self.x = 2

tk = Tk()
tk.title("Game")
tk.resizable(0, 0)
tk.wm_attributes("-topmost", 1)
canvas = Canvas(tk, width=500, height=400, bd=0,
highlightthickness=0)
canvas.pack()
tk.update()

paddle = Paddle(canvas, 'blue')
ball = Ball(canvas, paddle, 'red')

while 1:
    if ball.hit_bottom == False:
        ball.draw()
        paddle.draw()
    tk.update_idletasks()
    tk.update()
    time.sleep(0.01)
```

3.8 Primerjava programskih jezikov

V tabeli 3.1 so primerjani programski jeziki, ki sem jih pregledala v diplomski nalogi. Primerjala sem nekaj tehničnih podrobnosti, katere programske konstrukte lahko uporabljamo, stopnjo objektnega programiranja, način predstavitve kode, konstrukcijo projekta, na kakšen način preprečujejo sintaktične napake in način shranjevanja ter uvoza [18, 12, 11].

Prednost Scratcha za učence stare od 9 do 12 let na slovenskem prostoru je, da je dostopen v maternem jeziku, saj učenci pri tej starosti še ne obvladajo angleškega jezika in bi tako imeli težave pri razumevanju. Poleg tega omogoča uporabo vseh osnovnih programskih konstruktov. Primeren je tudi zaradi načina predstavitve kode in konstrukcije projekta, saj z bloki onemogočimo sintaktične napake ter omogočimo pregled nad možnimi ukazi. Prav tako je dobro poskrbljeno za preprečevanje napak v kodi in shranjevanje. Poleg tega je zelo priljubljena spletna skupnost za deljenje in objavljanje programov.

		LOGO	GREENFOOT	ALICE	KODU	MINDSTORMS	APPINVENTOR	PYTHON	SCRATCH
Tehnične podrobnosti	Leto nastanka	1967	2003	2000	2009	1994	2010	1990	2006
	Slovenščina	NE	NE	NE	NE	NE	NE	NE	DA
	Brezplačno	DA	DA	DA	DA	DA/NE*	DA	DA	DA
	Temelji na programskem jeziku	LISP	Java	Java	x	veliko	LOGO	x	SMaltalk - Squeak
	Starost otrok	6 - 9	10 – 18+	10 - 14	3 - 14	10 - 21	10 – 18	10 – 18+	8 - 12
	Funkcijsko/objektno	Funkcijsko	Objektno	Objektno	Objektno	Objektno	Objektno	Objektno in funkcijsko	Objektno
Programski konstrukti	Metode	DA	DA	DA	NE	DA	DA	DA	DA
	Funkcije	DA	NE	NE	NE	NE	DA	DA	DA
	Spremenljivke	DA	DA	DA	DA	DA	DA	DA	DA
	Argumenti	DA	DA	DA	NE	DA	DA	DA	DA
	Rekurzija	DA	DA	DA	NE	NE	NE	DA	DA
	Pogoji	DA	DA	DA	DA	DA	DA	DA	DA
	Zanka for	DA	DA	DA	NE	DA	DA	DA	DA
	Zanka while	DA	DA	DA	NE	DA	DA	DA	DA
Stopnja objektnega programiranja	Objekti	NE	DA	DA	DA	DA	DA	DA	DA
	Dedovanje	NE	DA	DA	DA	NE	DA	DA	NE
	Razredi	NE	DA	DA	NE	DA	DA	DA	NE
Predstavitev kode	Tekst	DA	DA	NE	NE	NE	NE	DA	NE
	Bloki oz. sestavljanke	NE	NE	DA	DA	DA	DA	NE	DA
	Ikone	NE	NE	NE	DA	DA	NE	NE	NE
Konstrukcija projekta	Tipikanje	DA	DA	MALO	MALO	MALO	MALO	DA	MALO
	Izbiranje	NE	NE	DA	DA	DA	DA	NE	DA
	Pravokotne kartice	NE	NE	DA	DA	DA	DA	NE	DA
	Ikone	NE	NE	NE	DA	DA	NE	NE	NE
Preprečevanje sintaktičnih napak	Ujemanje oblik	NE	NE	DA	DA	DA	DA	NE	DA
	Izbiranje ustreznih opcij	NE	NE	DA	DA	DA	DA	NE	DA
	Sinaktično usmerjeno urejanje	NE	DA	NE	NE	NE	NE	NE	NE
Shranjevanje in izvoz	Svoj format	.LGO	.gfar	.a3p	.kodu	.ev3	.apk	.py	.sb
	Shrani na internetu	DA	NE	NE	NE	NE	DA	DA	DA
	Skupnost	NE	DA	NE	DA	DA	NE	NE	DA

*potrebno je kupiti komplet Lego Mindstorms, aplikacija je brezplačna

Tabela 3.1: Primerjava programskih jezikov za učenje programiranja otrok

Poglavje 4

Scratch

4.1 Splošno o Scratchu

Scratch[1] je programski jezik, ki so ga leta 2007 razvili na inštitutu Media Lab. Ideja za nov programski jezik in okolje za otroke je izvirala iz že prej obstoječih programskih jezikov Logo in Etoys. Ustvarjen je bil z namenom, da se otroci z njim naučijo programiranja interaktivnih zgodb, iger in animacij. Na spletni strani lahko uporabniki svoje izdelke delijo s spletno skupnostjo ter si tam tudi ogledajo, kaj so ustvarili drugi [22].

Namenjen je spodbujanju kreativnega razmišljanja ter sistematičnega reševanja problemov. Ciljna skupina so otroci od 8. do 16. leta starosti, vendar ga lahko uporabljajo tudi mlajši ali starejši. V tem trenutku (26. 7. 2014) ima 3.680.484 registriranih uporabnikov, to število se v zadnjem letu še posebej hitro povečuje[2]. Dodatne pozitivne lastnosti Scratcha so tudi usmerjenost v oblikovanje, saj je ta aspekt pri učenju programiranja v srednjih šolah in na fakultetah velikokrat zanemarjen. Oblikovanje in ustvarjanje je omogočeno z uvažanjem različnih slik, zvokov, glasbe iz računalnika, interneta ali pa figure in ozadje narišejo kar otroci sami [27].

Programski jezik Scratch in njegovo okolje sta izredno atraktivna še posebej za mlajšo populacijo. Scratchev uporabniški vmesnik je razvit tako, da ima le

[1] `http://scratch.mit.edu/`

[2] `http://scratch.mit.edu/statistics/`

eno okno. Na njem lahko vidimo ukaze, ki jih lahko uporabimo, površino, na kateri se piše program, gumb za zaganjanje napisane kode ter prostor, kjer se izvede prikaz zagnane kode. Uporabniku prijazno je tudi to, da lahko vsak ukaz preizkusi z dvojim klikom, brez da bi za to moral pognati celotno že napisano kodo. Poleg tega so se, kot pri programskem jeziku Logo, potrudili omejiti vsa sporočila o napakah. Ukazi, ki si po pravilih programiranja ne morejo slediti, se že avtomatsko ne dajo sestaviti skupaj. To omogoča prijaznejšo uporabniško izkušnjo, saj otroci preizkušajo, kaj je prav in kaj narobe. Obvestila o napačni kodi ob zagonu programa bi jih v nasprotnem primeru lahko navdala z negativnimi občutki, če bi se napake pogosto pojavljale.

Programskih ukazov otroci ne pišejo sami, temveč so večinoma že napisani v blokih, ki so v obliki lego kock in jih lahko sestavljajo skupaj tako, da ukaze le povlečejo v prostor za programsko kodo. S tem je začetnikom olajšano učenje programiranja, saj pri tem ni težav s sintaktičnimi napakami [22].

Scratch ima svoje bloke razporejene v posebne skupine, da jih otroci lažje najdejo. Skupaj jih je 125, vendar se nekateri prikažejo šele, ko jih uporabnik potrebuje. Skupine blokov so: [15, 21]

- bloki za premikanje
  - V tej skupini najdemo ukaze za premikanje in obračanje ter podatke o trenutni lokaciji in smeri premikanja.
- izgled
  - Tu se nahajajo ukazi za prikazovanje govornih oblačkov, ukazi za skrivanje in prikaz figur, menjanje njihovega videza ter ozadja, spreminjanje barv, velikosti, prekrivanja pri prikazovanju elementov. Tu najdemo še podatke o trenutnem videzu, ozadju in velikosti figure.
- zvok
  - V skupini z zvoki so ukazi za predvajanje zvoka, spreminjanje glasnosti, hitrosti, izbire posameznega zvoka ter igranje posameznih not. Tu sta spremenljivki za glasnost in hitrost.
- svinčnik

  - V kategoriji svinčnik lahko figura uporablja pisalo in z njim riše, briše, odtisuje, nastavlja barvo, velikost, senco in širino svinčnika.

- podatki

  - V sekciji s podatki lahko uporabnik ustvari novo spremenljivko ali seznam. Ko ju ustvari, jima lahko spreminja ime, ju inicializira, prikaže ali skrije na grafičnem prikazu. Pri seznamu lahko spreminja samo določene elemente, pridobi dolžino seznama ter preverja ali vsebuje določen element.

- dogodki

  - V dogodkih se nahajajo ukazi za začetek izvajanja kode. Ti se vedno začnejo ob nekem dogodku: če kliknemo zastavico, pritisnemo želeno tipko, ko kliknemo figuro, se spremeni ozadje, poveča glasnost, spremeni video gib ali štoparica, ko prejme določeno sporočilo. V tej kategoriji imamo tudi ukaza za objavljanje sporočil in možnost njihovega poimenovanja.

- krmiljenje

  - V sklopu krmiljenje najdemo zanke in pogojne stavke. Poleg tega so tu tudi ukazi za čakanje, ustavljanje celotnega izvajanja ter ukazi za kloniranje figur.

- zaznavanje

  - V kategoriji zaznavanje se nahajajo pogoji v zvezi z dotikanjem določene figure, barve, razdalje do ostalih figur. Tu je tudi možnost postavljanja vprašanj in pomnjenja uporabnikovega odgovora. Poleg tega tu zaznava tudi pritisk tipk na tipkovnici ali pritisk miške. Nekaj je tudi ukazov za video zaznavanje gibanja, njegovo uporabo in nastavljanje prosojnosti. Poleg tega je v tej kategoriji možno dodati uporabo štoparice in njeno ponastavljanje, dobiti koordinate oddaljenosti od določene figure, čas izvajanja, število dni do leta 2000 in uporabniško ime uporabnika Scratcha.

- operatorji
  - Z operatorji se lahko uporablja matematične operacije, kot so: seštevanje, odštevanje, množenje, deljenje, ostanek, večje, manjše, je enako, zaokroževanje, korenjenje, generiranje naključnega števila. Poleg tega je na voljo še več zahtevnejših matematičnih funkcij, kot so: trigonometrične funkcije, absolutna vredost, logaritmiranje, uporaba eksponentne funkcije ... Tu se nahajajo tudi logični operatorji: in, ali ter negacija. Obstaja tudi možnost združevanja besedilnih vnosov ter dostopa do dela besedila.
- možnost ustvarjanja novih blokov
  - Tu se lahko ustvari nova funkcija, ki jo lahko nato uporabljamo kjerkoli v programu. Poleg tega se lahko doda tudi razširitev, ki omogoča povezovanje s PicoBoard ali LEGO WeDo.

Sintaksa jezika je tako sestavljena iz blokov, ki jih povlečemo na površino za pisanje kode. Oblike blokov nakazujejo, kateri bloki se z njimi lahko držijo in katere kombinacije niso mogoče. Vpisujemo večinoma le velikost kotov, koordinat, imena spremenljivk in podobne manjše ukaze. Da ni veliko sintaktičnih napak, ima kar nekaj ukazov tudi možnost izbire preko padajočega menuja.

Podatkovni tipi, ki jih lahko uporabljamo v Scrachu, so: logične vrednosti, števila in besedni nizi. Scratch temelji na objektno orientiranem programiranju, saj so figure objekti in vsako zaporedje ukazov se nanaša na določen objekt oz. na ozadje.

Interakcija med posameznimi figurami oziroma objekti ne poteka direktno. Namesto tega Scratch omogoča oddajo sporočil za komunikacijo in sinhronizacijo. Njegov model je odnos 'ena proti mnogo', saj figura pošlje sporočilo vsem ostalim in ne le izbrani figuri. To pomeni, da eno sporočilo lahko sproži dogodke več figuram [22].

Vzporednost ali večopravilnost je velikokrat uvrščena med težje programerske tehnike. Vendar za uporabnike Scratcha ni nenavadno, da lahko ena figura izvaja več ukazov naenkrat. Programi velikokrat izgledajo tako, da program naenkrat skrbi, da se figura odbija od roba, premika naprej in ob pritisku na določen gumb zraven še izvaja dodatno nalogo.

Scratch lahko brezplačno uporabljajo otroci na spletni strani[3] ali pa kot aplikacijo, ki si jo lahko prenesejo na svoj računalnik. Obe različici sta na voljo v slovenskem jeziku.

Eden izmed izredno uporabnih dodatnih funkcionalnosti Scratcha je skupnost. Scratch namreč omogoča deljenje projektov, ki jih otroci sami sprogramirajo ter njihov vpogled v druge. Ta trenutek (26. 7. 2014) jih je javno objavljenih kar 6.071.948. Tu lahko otroci dobijo ideje za svoje programe, vpogled v kodo drugih in v razmišljanje svojih vrstnikov. Poleg tega lahko celo spreminjajo programe ostalih, ne da bi pri tem uničili originalnega. Možno je komentiranje deljenih projektov, kar omogoča prenos idej, znanja in možnosti izboljševanja in medsebojne pomoči med otroki. Delno se učijo tudi določene stopnje kritičnosti in ocenjevanja svojih in ostalih programov. Možnost shranjevanja v studiih omogoča deljenje programov v določeni skupini, kar je še posebej uporabno za razredne skupnosti, krožke programiranja in podobne delavnice, ki v organizirani skupini razvijajo programe [29].

Do svojih izdelkov lahko s pomočjo spletne aplikacije preprosto dostopajo v svoji mapi, na svojem računalniku pa imajo izdelke shranjene lokalno.

Scratch je tako programski jezik, ki temelji na preprostosti, od grafičnega uporabniškega vmesnika, do poenostavljenih blokov z ukazi za programiranje. Primeren je za otroke, ki se s programiranjem še niso srečali in se lahko s preprostim, a dovršemin programskim okoljem in jezikom spoznajo z vsemi osnovnimi programskimi konstrukti in funkcijami, poleg tega pa ob tem pridobijo tudi nekaj matematičnega znanja [26].

4.2 Opažanja in ugotovitve s krožka programiranja in Poletne šole FRI

4.2.1 Primerjava krožka in tridnevne delavnice

Krožek programiranja v Scratchu je potekal na Osnovni šoli Alojzija Šuštarja v Ljubljani za osnovnošolce od 4. do 7. razreda. Vsak teden so reševali različne naloge in se s tem spoznavali s programiranjem in programskimi konstrukti.

[3] `http://scratch.mit.edu/`

Projekt Čarajmo v Scratchu se je odvijal kot tridnevna delavnica v sklopu Poletne šole FRI in je bil namenjen otrokom od 4. do 6. razreda osnovne šole.

Sami naravi krožka in poletne šole sta se zelo razlikovali po tipih nalog, načinu dela in razlage, kot tudi intenzivnosti spoznavanja s Scratchem. Na obeh projektih je bilo večina otrok začetnikov pri uporabi Scratcha. Krožek, ki je potekal pol leta, je bil zamišljen kot spodbujanje lastnega programiranja. Na začetku ure je bila navadno predstavljena naloga, ki so se je otroci po želji lotili z izdelavo, potrebovali pa so različne oblike pomoči. Nekateri več z osnovnim iskanjem ukazov, drugi z uporabo ukazov, najboljši pa so se ukvarjali z nadgradnjami in jih je zanimalo, če je nalogo mogoče sprogramirati na način, ki so si ga sami izmislili. Tem je bilo ponavadi potrebno pomagati razbiti problem na manjše dele, da so se ga lahko sistematično lotili.

Medtem ko je bilo na Poletni šoli FRI: Čarajmo v Scratchu drugače. Naloge so bile znane, nekatere so puščale ustvarjalnosti svojo pot, spet drugih smo se lotili skupaj in jih sprogramirali vodeno, tako da smo na koncu dobili vsi enak izdelek, ki so ga posamezniki na koncu še dodelali in nadgradili. Pri poletni šoli je bilo lažje sestaviti obsežnejše naloge, saj so tudi otroci imeli veliko več časa za programiranje, kot so ga imeli otroci na krožku, kjer so z nalogo morali zaključiti v 45 minutah. Ravno zato sem svoje naloge prilagodila tako, da niso prezahtevne in jih, predvsem na začetku, v eni uri ne bi mogli narediti.

Pozitivna stran krožka je bila, da so otroci po uri večkrat ostajali dlje, ker so želeli zaključiti z izdelavo, ali so se sami doma ukvarjali z dopolnjevanjem in razširjanjem nalog.

Pri obeh projektih se nismo usmerili direktno v to, da bi otroci razumeli, katere programske konstrukte uporabljajo, ampak v to, da smo jih z različnimi nalogami spodbudili k uporabljanju raznolikih konstruktov. To bi v prihodnje lahko izboljšali v smer zavedanja otrok, da uporabljajo programerske koncepte, ki niso več le igra.

4.2.2 Predznanje

Večina dela s Scratchem ne zahteva matematičnega, logičnega ali računalniškega predznanja. Vendar je nekaj matematičnih ukazov, ki se ves čas uporabljajo, in jih otroci vseh starosti v slovenskem prostoru še ne poznajo. Uporaba in razumevanje

kotov sta v učnem načrtu šele v šestem razredu. Do tega razreda poznajo otroci le pravi kot, tako se lahko kotov sami naučijo s poskušanjem ali pa jim mentor razloži njihovo definicijo.

Površina, na kateri so figure in se odvija njihovo premikanje, je v Scratchu predstavljena s koordinatnim sistemom, figure se po njem večinoma premikajo po koordinatah, lahko pa tudi po korakih. Učenje koordinatnega sistema je v učnem načrtu za sedmi razred, tako da ga v drugi triadi osnovne šole še ne poznajo dobro in je zato potrebna bolj natančna razlaga [8].

4.2.3 Dojemanje konstruktov

Otroci konstrukte dojemajo zelo različno. Abstraktno razmišljanje naj bi se razvilo nekje po 12. letu starosti, kar je primerno obdobje za začetek učenja programiranja [24].

Nekateri otroci kode ne razumejo. Potrebno je veliko pomoči in vodenja. Njim je potrebno nalogo razdeliti na majhne dele, da jo lahko rešijo. Drugi otroci že razumejo osnovne ukaze, tudi kodo drugih bi lahko prebrali in jo delno razumeli, vendar ne razumejo bistva, kako celoten program deluje. Potek programa lahko povejo le za vsak ukaz posebej, kaj se z elementom dogaja, vendar ne razumejo celotnega konteksta programa.

Uspešnejši otroci imajo že nekaj več razumevanja in za nekatere dele programa že razumejo, kako se odziva, zato lahko predvidijo, kaj se bo zgodilo, če preberejo kodo. Ustavi se jim pri težjih in bolj zapletenih konceptih, predvsem, če so sestavljeni iz več ukazov. Ti sicer potrebujejo manj pomoči kot ostali, vendar še vedno ne razumejo vsega.

Nekaj otrok naloge rešuje in bere samostojno. Zaradi dovolj izkušenj in znanja lahko sklepajo, kaj bo program naredil in razumejo tudi težje in sestavljene naloge. To znanje predvsem pridobijo z veliko programiranja, poskušanja in spreminjanjem tuje kode [6].

4.2.4 Samoiniciativa in samostojnost pri reševanju

Ravno zaradi spodbudnega okolja Scratch je bilo pri otrocih v obeh projektih opaziti ogromno samoiniciativnosti. Najprej so reševali naloge, ki smo jim jih

predlagali mi, nato so se sami lotili nadgradenj ali pa so ideje dobili v deljenih projektih v skupnosti na Scratchevi spletni strani. Okolje jih pritegne vizualno, na voljo jim da bloke, ki so izredno dobro razumljivi, da vsak ve, kaj se bo zgodilo ob kliku nanj.

Zanimivo je bilo opazovati tudi to, da večinoma niso uporabljali enih in istih blokov, ampak so se trudili, da bi uporabili čim več različnih in spoznali vse, kar bi povečalo interaktivnost njihove igre ali animacije, v bistvu pa so s tem širili tudi poznavanje programskih konstruktov in ukazov.

Samostojnost pri reševanju je bila pri obeh projektih prvo uro malo manjša, saj so se spoznavali z okoljem, vendar po začetnem spoznavanju ni bilo potrebno veliko pomoči pri iskanju blokov, saj so ti urejeni v logične enote in jih ni težko poiskati. Z dobro organiziranim okoljem za programiranje samostojnost pri delu le še raste.

4.3 Komentar nalog, uporabljenih v Nalogah za krožek računalništva

Naloge, ki sem jih pripravila za krožek računalništva, so namenjene učiteljem in staršem, ki bi svoje otroke želeli naučiti programiranja. Njihovo predznanje ni pomembno, saj so naloge razložene tako, da jih lahko razume vsak.

V uvodu sem predstavila, komu je knjižica namenjena, kaj je Scratch in katere programske koncepte nameravam naučiti otroke skozi naloge.

V poglavju Uporaba Scratcha sem slikovno in besedno opisala postopek uporabe spletne strani, prijavo in registracijo v program ter ustvarjanje novega projekta.

4.3.1 Zaporedje ukazov

Prvi programski konstrukt, ki se ga otroci naučijo s pripravljenimi nalogami, je zaporedje ukazov. V rumenem okvirju je pred nekaterimi nalogami predstavljen

nov programski konstrukt, ki ga otroci skozi nadaljnje naloge spoznajo. Navadno z nekim primerom iz njihovega življenja, da ga lažje razumejo.

Sledijo naloge, ki sodijo pod to poglavje. Vse naloge so podane na enak način. Najprej slika približno prikaže, kakšna je postavitev in tematika naloge. Sledijo navodila. Oboje je v modrem okvirčku, saj ta del lahko vidijo tudi otroci. Nato so zapisani potrebno predznanje in cilji, ki jih želimo s to nalogo doseči. Navedena je tudi povezava na materiale, to so deljeni projekti na Scratchu, ki sem jih ustvarila v ta namen. Učitelj naj bi otrokom vedno po podanih navodilih pokazal, na kakšen način naj bi program deloval, vendar jim ne sme razkriti programske kode za naloge. Pri poteku podajanja nalog je po korakih opisano, kako otrokom razložiti, kaj naj naredijo v naslednjem koraku, kje naj najdejo ukaze in na kakšen način naj jih uporabijo. Ponekod so priloženi zaslonski posnetki, da bi olajšali iskanje po spletni strani in se raje osredotočili na programiranje samo.

V sklopu Zaporedje ukazov najdemo preproste naloge. Navodila so podana tako, da so že v pravem zaporedju, kot si sledijo ukazi. Ukazi, ki jih otroci pri tem uporabljajo, so predvsem za premikanje in izgled.

Pri drugi nalogi z zaporedjem se naučijo prve uporabe spreminjanja kotov, ki jih je najmlajšim učencem potrebno razložiti, saj znanje, ki ga dobijo do 5. razreda osnovne šole navadno ni dovolj.

Tretja naloga že vsebuje več ukazov s premikanjem in dobro poznavanje koordinatnega sistema. Tudi to znanje je v 4. in 5. razredu še pomanjkljivo, zato je tudi tu potrebna dodatna matematična razlaga [8]. Še vedno se naloge izvajajo v zaporedju. Od ukazov se uporabljajo tisti s področja premikanja in čakanje.

Večjih težav s spoznavanjem in uporabo zaporednih ukazov naj otroci ne bi imeli, saj se bloki že sami sestavljajo na način, da se skladajo drug pod drugega in po vrsti izvajajo ukaze. To se otrokom zdi smiselno in logično, tako da tu ne pričakujem težav z razumevanjem.

4.3.2 Zanke

Naslednji programski koncept so zanke, ker nekako logično sledijo učenju programiranja po spoznavanju z zaporednim izvajanjem programa. Z zankami si otroci olajšajo delo ali omogočijo neprestano premikanje nekega elementa. Pri nalogi Izgubljena žoga se spoznajo z neskončno zanko, kasneje pa s končno zanko, ko rišejo

določeno število stopnic tako, da ukazov ne pišejo večkrat, ampak se izvedejo v zanki tolikokrat, kolikor sami želijo.

Poleg teh dveh tipov zank se spoznajo še z enim: naj se ukazi v zanki ponavljajo toliko časa, dokler je izpolnjen nek pogoj.

V Scratchu so zanke narejene tako, da ukaze vstavljamo v zanko in je s tem narejen zamik, da je vidno, kateri ukazi se izvajajo v zanki. Po neskončni zanki ni možno dodajati ostalih ukazov, po vseh ostalih zanke lahko normalno nadaljujemo z izvajanjem programa.

Tudi v tem poglavju so naloge zelo osnovne in usmerjene v razumevanje zank. Tako koda ni predolga in zahtevna, saj bi s tem le preusmerila pozornost od uporabe zank. Navodila so napisana tako, da otrok začuti potrebo po tem, da uporabi zanko, saj se brez nje program ali ne more izvajati ali pa je kode preveč in se prevečkrat ponovi in jo uporaba zanke očitno optimizira.

4.3.3 Pogojni stavki

Prva naloga s pogojnim stavkom je osnovni stavek z enim pogojem in ukazom, ki se izvede, če je le-ta izpolnjen. Drugače se izvede drugi ukaz. Druga naloga je podobna, le da v pogoj vstavimo matematični znak za primerjavo velikosti števil. Sledi naloga, ki je sestavljena iz brezpogojne zanke, več ukazov in pogojnega stavka, saj otroci do sedaj znajo že dovolj, da izdelajo enostavno igro in kombinirajo znanje, ki so ga pridobili do sedaj.

Pri učenju pogojnih stavkov ni pričakovati veliko težav, saj je slovenski prevod razumljiv in je logično, kako se bloke uporablja.

Težava, ki se pojavi v nadaljevanju in na sploh pri programiranju v Scratchu, je razumevanje otrok, da se pogojni stavek preveri le enkrat in je za neprestano preverjanje pogojnega stavka potrebno uporabiti neskončno zanko.

Če želimo namreč preverjati, ali se je neka figura dotaknila druge, moramo ukaz za preverjanje dotika dati v neskončno zanko in vanjo preverjanje dotika, kot je prikazano na sliki 4.1. Drugače se ukaz, ki preverja, ali sta figuri dotaknjeni, izvede le enkrat.

Slika 4.1: Preverjanje dotika v neskončni zanki

4.3.4 Odziv na dogodke

Najprej se otroci z nalogami naučijo odzivati na vnose s tipkovnice. Figura postavi vprašanje in uporabnik mora vpisati odgovor. Odgovor se shrani v spremenljivko odgovor, vendar pri tej nalogi še ne poudarimo, da je to spremenljivka, čeprav jo tako že uporabljamo.

Pri nalogi Labirint, ki zahteva dobro predznanje iz prejšnjih nalog in tudi drugih konstruktov, uporabljamo še več odzivanja na uporabnikove ukaze. Glede na tipke, ki jih uporabnik pritisne, se premika miš in tako odziva na dogodke.

Tu otrok podzavestno uporabi tudi večnitnost, vendar je ta podrobneje razložena pri drugi nalogi.

Vnos s tipkovnico se pojavi šele pri vprašanju, ki ga postavi figura ali celoten program. Ta ukaz bi sami otroci morda malo težje našli, vendar so ga tako prisiljeni poiskati zaradi tipa naloge. Tudi uporaba spremenljivke odgovor ne povzroča velikih težav, vendar je potrebno poudariti, da si Scratch zapomni samo zadnji odgovor, tako bi morali za več vprašanj in odgovorov le-te vmes shranjevati v svoje spremenljivke.

Manj težav imajo otroci z uporabo odziva na pritisnjene tipke, saj so ukazi za pritisk tipke narejeni kot dogodek, ki posluša, kdaj bo želena tipka kliknjena.

4.3.5 Spremenljivke

Spremenljivke so otrokom pri srcu za štetje dobljenih ali izgubljenih točk pri igranju iger. Najprej jih ustvarijo, vendar jim tipa spremenljivke ni potrebno določiti,

lahko izberejo, ali bodo globalne ali lokalne. Tako je sestavljena naloga v tem poglavju. Najprej je potrebno spremenljivko ustvariti, jo inicializirati in ob nekem dogodku se ji spremeni vrednost.

Uporabo spremenljivk in njihov prikaz je na začetku malce težje razumeti, vendar spremenljivke otroci radi uporabljajo in po prvem srečanju z njimi hitro razumejo, kaj pomenijo in kako se spreminjajo. Potrebno jim je le razložiti, da naj se ob vsakem naslednjem začetku igre postavi število na začetno vrednost, potem pa nimajo večjih težav.

4.3.6 Večnitno izvajanje

Večnitno izvajanje so otroci sicer nezavedno uporabljali že prej, vendar se pri naslednji nalogi s tem srečajo še bolj poudarjeno, da razumejo, kako se stvari lahko odvijajo istočasno.

Otrokom samo sestavljanje večnitnih ukazov ne dela preglavic, vendar je za razumevanje samega programskega konstrukta to potrebno poudariti še v samostojni nalogi.

4.3.7 Sestavljene naloge

Preostale naloge so sestavljene. Uporabljajo večino programskih konstruktov, ki so jih otroci spoznali skozi reševanje nalog in so bolj zanimive, saj so v večini igre, ki otroke pritegnejo k programiranju.

Tabele

Med sestavljenimi nalogami najdemo tudi nalogo s tabelami, kjer kmet vsakič, ko se dotakne določene živali, to doda na seznam živali. Uporaba tabel je podobna kot pri spremenljivkah. Najprej jo ustvarimo, nato pa lahko dodajamo, brišemo, menjamo in dostopamo do posameznih elementov v tabeli.

Uporaba tabel ni tako samoumevna, kot so drugi programski konstrukti. Potrebne so predvsem pri nalogah in igrah z veliko podatki, ki si jih je potrebno zapomniti. Sama uporaba tabel pa je podobna kot pri spremenljivkah in naj ne bi povzročala večjih preglavic.

4.3.8 Sporočila

Naloga s sporočili je narejena tako, da po naključno iztečenem času objavi sporočilo, da lahko avtomobila začneta tekmovati. Ko ta dva prejmeta sporočilo, se začneta premikati.

Z uporabo pošiljanja in prejemanja sporočil figure med seboj uporabljajo interakcijo in s tem skrbijo za sinhronost izvajanja.

Menim, da je sporočila najtežje razumeti, saj je v začetku težko razumeti, kako bo druga figura lahko 'slišala' objavo nekega sporočila in ni tako logično, kot se zdi po nekajkratni uporabi.

4.3.9 Dodatne naloge

Predlog dodatnih nalog, ki niso zapisane v sklopu diplome, so razne igre in naloge iz drugih programov učenja programiranja, kot je na primer Vidra[4]. Ker je neprestano sedenje za računalnikom za otroke nezdravo in težavno, je pomembno, da vmes njihovo pozornost pridobimo z raznimi dinamičnimi igrami, ob tem pa lahko izvejo še kakšen podatek računalniške tematike.

Poleg računalništva brez računalnika predlagam tudi popravljanje ali nadaljevanje tuje kode, saj se s tem otroci učijo brati ukaze drugih, dojemati njihovo razmišljanje, dobijo vpogled v dobro in slabo kodo, dobijo ideje za optimizacijo. S tem pa tudi preverimo, ali otroci svojo kodo vstavljajo le s poskušanjem ali kodo pišejo tako, da jo resnično razumejo. Preizkusimo jih ali znajo popraviti napake, ki se programerjem velikokrat pojavijo in ali jih znajo na pravilen način popraviti [13].

4.3.10 Ostali programski konstrukti

Scratch podpira tudi pisanje funkcij, vendar v teh nalogah niso razložene. To bi rada pokazala v drugem delu knjižice. Mogoča je tudi uporaba rekurzije, za katero menim, da je za povprečne otroke v 4. in 5. razredu še nekoliko pretežka za razumevanje.

V Scratchu ni možno definirati lastnih razredov objektov, uporabljati dedovanja ter brati in pisati iz datoteke. Vendar to niso tako pomembni osnovni kon-

[4] `http://vidra.fri.uni-lj.si/`

strukti, ki bi jih otroci morali razumeti. Tako je znanje, ki ga dobimo z nalogami, zadovoljivo za učence druge triade osnovne šole.

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo si najprej ogledali programske jezike in okolja za otroke, jih malce bolj podrobno spoznali, videli, kako se v njem programira in piše koda, poleg tega smo jih tudi primerjali med seboj in s programskim jezikom Scratch, ki je glede na ostale programske jezike najbolj primeren za zgodnje učenje programiranja. To omogoča zaradi enostavne interakcije z bloki, ki olajšajo pisanje kode, grafičnega vmesnika, skupnosti, ki nudi deljenje in komentiranje izdelkov z ostalimi uporabniki ter same osredotočenosti na osnovne in nezahtevne programske konstrukte.

Sledila je kratka predstavitev trenutnega stanja z učenjem programiranja v slovenskih osnovnih šolah in predstavitev možnosti za izboljšanje, ki že poteka.

V nadaljevanju smo si pogledali, kaj sploh je Scratch in kaj omogoča.

Sledili so komentar in ugotovitve z mojih opazovanj pri krožku in poletni šoli, ki so obsežnejši v prilogi z naslovom Poročilo s krožka programiranja in Poletne šole FRI Čarajmo v Scratchu.

Na koncu smo lahko prebrali še utemeljitev izbire nalog, ki sem jih uporabila v prilogi Naloge za krožek računalništva.

Naloge so bile narejene tako, da bodo uporabne širšim množicam, javno bodo objavljene in deljene v obliki .pdf. Poleg tega se že uporabljajo pri neobveznem izbirnem predmetu v osnovni šoli. Nekaj od njih smo jih uporabili že na Poletni šoli FRI in podobnih dogodkih, namenjenih poučevanju programiranja otrok.

Naloge so namenjene predvsem učiteljem in staršem, ki se želijo ukvarjati z otroki in jih naučiti osnov programiranja in programskih konstruktov ne glede

na njihovo predznanje. Otroci z nalogami tako pridobijo dovolj znanja, da sami kasneje ustvarjajo brez pomoči staršev, saj jim naloge omogočijo spoznavanje z vsemi bloki in funkcijami, ki jih nudi Scratch.

V nadaljevanju imam namen izdati še naslednji del knjižice, ki bi otroke skozi ustvarjanje iger pripeljal do uporabe sortirnih algoritmov in zahtevnejših programov. Namenjen bi bil nadaljevanju krožka za vse tiste, ki Scratch že poznajo in so ga navajeni uporabljati.

Literatura

[1] *About the Greenfoot.* Dostopno na: http://www.greenfoot.org/about/contributors.html.

[2] *Alice.* Dostopno na: http://www.alice.org/index.php?page=what_is_alice/what_is_alice.

[3] *Alice Programing.* Dostopno na: http://computerkiddoswiki.pbworks.com/w/page/16304640/Alice%20Programing.

[4] *Alice: Teaching Programming through 3D Animation and Storytelling.* Dostopno na: http://www.cmu.edu/corporate/news/2007/features/alice.shtml.

[5] *Computer Science K–8: Building a Strong Foundation.* Dostopno na: https://csta.acm.org/Curriculum/sub/CurrFiles/CS_K-8_Building_a_Foundation.pdf.

[6] *Computing at School International comparisons.* Dostopno na: http://www.computingatschool.org.uk/data/uploads/internationalcomparisons-v5.pdf.

[7] *Mindstorms EV3.* Dostopno na: http://cache.lego.com/upload/contentTemplating/Mindstorms2News/otherfiles/download69B946250150B6AEBE7D48A0EE1A0595.pdf.

[8] *Učni načrt matematika.* Dostopno na: http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/prenovljeni_UN/UN_matematika.pdf.

[9] *Učni načrt računalništvo - neobvezni izbirni predmet.* Dostopno na: http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/os/devetletka/program_razsirjeni/Racunalnistvo_izbirni_neobvezni.pdf.

[10] *What is Logo?* Dostopno na: http://el.media.mit.edu/logo-foundation/logo/index.html.

[11] *Xylophone, AppInventor tutorial.* Dostopno na: http://www.appinventor.org/apps/xylophone/xylophone.pdf.

[12] Matthew MacLaure Allan Fowler, Teale Fristce. *Kodu Game Lab: a programming environment*, 2012. Dostopno na: http://tcjg.weebly.com/uploads/9/3/8/5/9385844/whitsun2012_fowler_et_al.pdf.

[13] K Brennan, M Chung, and J Hawson. Creative computing: A design-based introduction to computational thinking. *Retrieved May*, 9:2012, 2011.

[14] Jason R Briggs. *Python for kids: A playful introduction to programming.* no starch press, 2013.

[15] M Drahošová. Úvod do programovania v prostredí Scratch, 2010.

[16] Mr. Giansante. *Alice 3D Programming*, 2009. Dostopno na: http://www.lincoln.edu.ar/comp/alice/Alice.pdf.

[17] Jim Gindling, Andri Ioannidou, Jennifer Loh, Olav Lokkebo, and Alexander Repenning. Legosheets: a rule-based programming, simulation and manipulation environment for the lego programmable brick. In *Visual Languages, Proceedings., 11th IEEE International Symposium on*, pages 172–179. IEEE, 1995.

[18] Monika Gujberová and Peter Tomcsányi. Environments for programming in primary education. In *Informatics in schools: local proceedings of the 6th International Conference ISSEP 2013; selected papers; Oldenburg, Germany, February 26–March 2, 2013*, volume 6, page 53. Universitätsverlag Potsdam, 2013.

[19] Michael Kölling. The Greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):14, 2010.

[20] Peter Krebelj. *Python 3 za začetnike: vse kar potrebujete, da boste postali ython programerji*. Atelje Doria, 2013.

[21] Sonja Lajovic. *Scratch: nauči se programirati in postani računalniški maček*, 2011.

[22] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The Scratch Programming Language and Environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15, November 2010.

[23] Tim Peters. The zen of Python. In *Pro Python*, pages 301–302. Springer, 2010.

[24] Jean Piaget and Brbel Inhelder. *The growth of logical thinking from childhood to adolescence: An essay on the construction of formal operational structures*, volume 84. Routledge, 2013.

[25] Mitchel Resnick. All I Really Need to Know (About Creative Thinking) I Learned (by Studying How Children Learn) in Kindergarten. In *Proceedings of the 6th ACM SIGCHI Conference on Creativity &Amp; Cognition*, C&C '07, pages 1–6, New York, NY, USA, 2007. ACM.

[26] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[27] J. Sorva, J. Lönnberg, and L. Malmi. Students' ways of experiencing visual program simulation. *Computer Science Education*, 23(3):207–238, 2013.

[28] Ian Utting, Stephen Cooper, Michael Kölling, John Maloney, and Mitchel Resnick. Alice, Greenfoot, and Scratch – A Discussion. *Trans. Comput. Educ.*, 10(4):17:1–17:11, November 2010.

[29] Nicole Forsgren Velasquez, Deborah A. Fields, David Olsen, Taylor Martin, Mark C. Shepherd, Anna Strommer, and Yasmin B. Kafai. Novice programmers talking about projects: What automated text analysis reveals about online Scratch users' comments. In *Proceedings of the 2014 47th Hawaii International Conference on System Sciences*, HICSS '14, pages 1635–1644, Washington, DC, USA, 2014. IEEE Computer Society.

[30] David Wolber, Hal Abelson, Ellen Spertus, and Liz Looney. *App Inventor*. "O'Reilly Media, Inc.", 2011.

Dodatek A

Poročilo s krožka programiranja in Poletne šole FRI: Čarajmo v Scratchu

Poročilo s krožka programiranja in Poletne šole FRI: Čarajmo v Scratchu

Petra Mihalič

petra.mihalic92@gmail.com

Priloga k diplomski nalogi 2013/14
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

22. avgust 2014

Povzetek

Krožek programiranja v Scratchu je bil namenjen osnovnošolcem od 4. do 7. razreda. Vsak teden so reševali različne naloge in se preko tega spoznavali s programiranjem in programskimi konstrukti. Kljub različnemu predznanju so na krožku otroci sledili pripravljenim nalogam in jih glede na zanimanje tudi reševali ter tako napredovali v dojemanju konceptov. Zaradi fleksibilnejšega programa je bilo opaziti, da se dekleta raje ukvarjajo s pripovedovanjem zgodb v Scratchu, fantje pa programirajo igre. Projekt Čarajmo v Scratchu se je odvijal kot trodnevna delavnica. Namenjen je bil otrokom od 4. do 6. razreda osnovne šole. Tu so bili otroci bolj omejeni na programiranje vnaprej določenih programov, ki so bili kompleksnejši, saj so imeli veliko več časa.

Ključne besede: Programiranje, otroci, Scratch, poučevanje, programski koncepti, krožek

1 Krožek programiranja v Scratchu

Krožek programiranja v Scratchu je potekal od oktobra 2013 do februarja 2014 vsak petek na Osnovni šoli Alojzija Šuštarja. Obiskovalo ga je od 10 do 15 otrok, od 4. do 7. razreda osnovne šole. Krožek je vodil profesor Janez Demšar. Pedagoško je sodelovala Irena Demšar, ki je otrokom pomagala pri razumevanju še ne poznanih matematičnih konceptov, kot so na primer koordinatni sistem in koti, saj tega znanja računalnikarji nimamo. Moja naloga je bila, da otrokom pomagam pri reševanju nalog in opazujem, na

kakšen način dojemajo programske konstrukte in na kakšen način se lotijo reševanja nalog. To opazovanje mi je pomagalo tudi kasneje pri sestavljanju nalog za učenje programiranja v Scratchu.

1.1 POROČILO 1

DATUM: 25. 10. 2013

ORODJE: Programski jezik Scratch

NALOGA: Naj neka figura izriše stopnice, se povzpne po njih navzgor, tam poskoči in skoči z vrha stopnic navzdol, ob tem spodaj dobi buško.

NAMEN: Uporaba grafične površine, orientiranje in premikanje po njej s spreminjanjem koordinat in kotov, premikanje elementov po površini. Uporaba svinčnika in njegove funkcije dvigni-spusti. Menjava vizualnega izgleda elementov.

UGOTOVITVE: Otroci so se pred mojim opazovanjem za eno uro že srečali s programskim jezikom Scratch, tako da so osnovne funkcije in vizualni uporabniški vmesnik že spoznali. Za večino otrok naloga ni bila tako zahtevna.

Nekaj mlajših otrok je imelo težave s koordinatnim sistemom in koti. Malce starejša dekleta so hitro ugotovila, kako se je potrebno premikati, da narišemo stopnice. Nekaterim je gospa Irena vizualno kar na tleh pokazala, kako se je potrebno obračati, da narišemo stopnice in so si tako lažje predstavljale premike in spreminjanje kotov.

Jaz sem večino časa preživela z dekletom, ki je želelo nalogo rešiti točno po navodilih. Najprej sva narisali stopnice in konjička, ki je skočil s stopnic. Potem sva po korakih popravljali in dodajali še ostale elemente. Z majhnimi namigi in preizkusi sva ugotovili, kako konjiček poskoči navzgor in skoči nazaj na isto mesto. Pregledovanje možnih že napisanih funkcij otrokom olajša izbiro in odločanje. Predvsem je v tem jeziku prijetno tudi preizkušanje posameznih elementov in funkcij, saj na njih dvakrat kliknemo in hitro ugotovimo, kaj le-ti naredijo. Ta lastnost jezika je prav prišla pri menjanju kostumov in elementov (konjička brez in z buško). Vmes je nato deklica vstavila tudi nekaj ukazov "počakaj", da je bilo videti, kako se program izvaja.

1.2 POROČILO 2

DATUM: 8. 11. 2013

ORODJE: Programski jezik Scratch

NALOGA: Imamo princa in princeso. Princ se ves čas obrača proti princesi. Kamor prestavimo princeso, se tako princ vedno obrne proti njej in se ob tem tudi premika v smer, v katero je obrnjen. Ko sta princ in princesa dovolj blizu skupaj, princesa zardi.

NAMEN: Uporaba hkratnega izvajanja dogodkov. Menjava izgleda elementov, premikanje po površini, obračanje v določeno smer, uporaba zanke.

UGOTOVITVE: Otroci so si najprej ogledali izvajanje kode programa. Nato jim je bilo potrebno postopoma razložiti, kaj se zgodi z vsako figuro posebej v naravnem jeziku. Iz tega so tako lahko sklepali, kako sprogramirajo program. Tokrat je bila večina učencev nekoliko starejših, tako da večjih težav niso imeli. Profesorjeva razlaga jim je bila zelo v pomoč, saj je zelo natančno razložil, kaj se zgodi s figurami in kako se odzivajo na nekatere dogodke. Še največ težav so imeli z vstavljanjem pogoja v zanko: približuj se, dokler nisi na neki razdalji in tedaj zamenjaj kostum elementu (deklica naj zardi). Neka deklica je imela kodo napisano na nekoliko nepravilen način, saj se je princ do njene princeske približeval le po navidezni linearni črti proti točno določeni koordinati, kjer je bila princeska. Želela sem, da bi kodo popravili, da bi delovala tudi, ko bi se princeska premikala, vendar sem jo na ta način le zmedla. Sama si je nalogo drugače zamislila in morali bi začeti od začetka, saj si je nalogo interpretirala na svoj način, ki pa ni bil tak kot zahtevan. Ob koncu sva ji nato s profesorjem na hiter način pomagala priti do rešitve.

Vmes sta Mark in Katarina sama poizkušala sprogramirati program. Katarina si je zamislila labirint, v katerem bi bilo nekaj žog in pajek, ki naj bi se tem žogam izogibal. Priprava grafičnega vmesnika ji ni delala težav, nato pa sta z Markom sprogramirala program s pomočjo definiranja in uporabe funkcij. Za pajka sta sprogramirala, da se premika glede na to, katero tipko pritisnemo (levo, desno, gor, dol). Uporabila sta celo spremenljivke.

Profesor je pri opazovanju Marka, ki je prej programiral svoj program, opazil zanimivo "napačno" razmišljanje pri uporabi funkcije. Želel je namreč napisati svojo funkcijo, ki bi delovala kot pogoj, vendar je noter vstavil pogojni stavek ravno tak, kot bi ga lahko uporabil namesto na novo napisane funkcije.

1.3 POROČILO 3

DATUM: 15. 11. 2013

ORODJE: Programski jezik Scratch

NALOGA: S pomočjo ugnezdenih zank nariši kvadrat. Nato naj se figura zamakne in ponovno izriše kvadrat ... Uporabi funkcijo kvadrat in jo vstavi v program namesto vsakokratnega risanja kvadrata.
Dodatno: nariši večkotnik in ga enako sukaj okoli enega izmed kotov.

NAMEN: Uporaba ugnezdenih zank, definiranje funkcij

UGOTOVITVE: Profesor je v začetku ure pokazal kar nekaj različnih in atraktivnih izrisov različnih elementov na površino. Otroci so takoj postali motivirani in so se hitro odločili, da bi radi sprogramirali tisti izris, ki se jim je zdel lep, a še vedno dovolj enostaven.
Najprej je bila njihova naloga, da s pomočjo zanke narišejo kvadrat. Razloženo jim je bilo, da naj najprej narišejo črto, se obrnejo za 90 stopinj in to ponovijo na tak način, da bodo dobili kvadrat. Večina otrok je to hitro razumela, najmlajšim je bilo to potrebno pokazati še v 'realnem prostoru'. Za tem so morali figuro zasukati za npr.: 15 stopinj in ponovno izrisati kvadrat ter tudi te premike in izrise dati v zanko. Nalogo so hitro razumeli in sprogramirali.
V nadaljevanju so se seznanili s pisanjem funkcij. Namesto da bi v zunanji zanki vsakič risali kvadrat iz črt in obrata, so ta del programa vstavili v funkcijo "kvadrat" in tako napisano funkcijo potem uporabili pri risanju sukajočih se kvadratov. Nadgradnja tega programiranja je bila, da naj bi izrisali trikotnik, šestkotnik ali nek večkotnik. Vendar jim koti niso bili eksplicitno podani. Profesor je razložil le, da se vedno, ko imamo nek lik in ga rišemo, obračamo tako, da na koncu naredimo poln obrat, to je 360 stopinj. Iz tega so otroci v petem razredu kar sami sklepali, da je kot pri trikotniku potemtakem velik 360/3 stopinj. Tudi te like so potem na svojo željo sukali za določen kot in risali lepe vzorce, vmes so se igrali tudi z vizualnimi učinki, kot je na primer spreminjanje barve. Profesor je v vmesnem času Katarini in njeni prijateljici želel predstaviti programiranje z rekurzijo. Risanje Kochove snežinke je bilo primerno ravno za to učenje. Deklici sta najprej s profesorjem na tablo skicirali nalogo in tudi napisali psevdokodo. Za tem so s pomočjo profesorja zapisali tudi program v Scratchu, vendar s profesorjem nisva mogla ravno ugotoviti, kako dobro sta deklici razumeli rekurzijo.

1.4 POROČILO 4

DATUM: 22. 11. 2013

ORODJE: Programski jezik Scratch

NALOGA: Otrokom je bila najprej predstavljena naloga: Kateri je pravi? V stolpcu so bili postavljeni trije enaki krožci. Na začetku se je en obarval rumeno in potem spet nazaj na tako barvo, da so bili vsi trije enaki. Nekaj časa so se ti krožci menjali med sabo, nato pa je uporabnik moral klikniti, kateri krožec je bil sprva označen. Če je kliknil pravilno, se je izpisal napis: "Bravo!"če pa napačno, se je izpisalo: "Narobe."
Druga naloga je bil labirint, ki je otrokom dobro poznan. Potrebno je bilo pripraviti ozadje oziroma stene labrinta, ki jih je bilo potrebno narisati. V nadaljevanju naj bi otroci pripravili še miško, ki teka po labirintu, in cilj, do katerega naj bi miška prišla. Za najhitrejše smo dodali še dodatno nalogo: naj premikanje po labirintu ovira še kakšna figura.

NAMEN: Uporaba spremenljivk, naključnih števil, koordinatnega sistema, neskončnih zank in pogojnih stavkov, odzivanje na dotik.

UGOTOVITVE: Prve naloge ni programiral nobeden izmed otrok. Predvidevam, da jim ni bila zanimiva. Bolj zabavna se jim je zdela za igranje in ugotavljanje, vendar jim je ni bilo izziv sprogramirati.
Drugo nalogo so reševali vsi. Risanje sten labirinta ni delalo težav, vendar je bilo kasneje nekaj problemov s tem, da so bili elementi preveliki in so bile stene podobnih barv kot ozadje. Ko smo popravili prve nevšečnosti, so ob vodenju profesorja otroci sprogramirali, kako naj se mišek odziva na klik levo, desno, gor in dol. Še enkrat je bilo potrebno obnoviti znanje kotov, vendar to ni delalo večjih težav. Sledila je realizacija zaznavanja zidov v labirintu, ki so jo z nekaj poskušanja vsi razumeli. Ob dotiku z drugo figuro se je igra zaključila in izpisala se je čestitka za zmagovalca.
Naloga je časovno kar precej zahtevna, tako da jo je težko od začetka do konca sprogramirati v eni uri. Dva sta imela že od prej narejen prvi del te naloge in sta se lahko tako osredotočila na dodatne ovire in figure na poti do cilja.
Naloga je ena izmed boljših, saj otroci vedo, kaj morajo realizirati, si to predstavljajo in želijo dokončati igro, da jo kasneje lahko igrajo. Vmes si še sami včasih dodajo kakšno dodatno nalogo s štetjem ali odštevanjem točk, pobiranjem življenj in podobno.

1.5 POROČILO 5

DATUM: 6. 12. 2013

ORODJE: Programski jezik Scratch

NALOGA: Animiraj zgodbo o Povodnem možu.
Druga naloga: Brani mačka pred pomarančami. V središču zaslona je maček in proti njemu letijo pomaranče. Če se ga pomaranča dotakne, izgubi točko. Na začetku naj ima uporabnik 10 točk, ki se odštevajo, če mačka zadane pomaranča. Narediti je potrebno še lopar, ki odbija pomaranče stran od mačka.

NAMEN: Pripovedovanje zgodb, uporaba domišljije, prenašanja sporočil, vnašanje odgovorov preko tipkovnice, uporaba spremenljivk, štetje točk, uporaba kotov.

UGOTOVITVE: Večina deklet se je osredotočila na pripovedovanje zgodb. Predlog je bil, naj animirajo zgodbo o Povodnem možu, ki bi jo nato lahko uporabili na prireditvi ob kulturnem prazniku. Dekleta so bila nad idejo navdušena in so se same, večinoma brez pomoči, igrale in spoznavale bloke ter jih uporabljale. Zanimivo se mi je zdelo to, da se je večina trudila, da bi uporabila čim več najrazličnejših blokov in ne samo tistih, ki so jih poznale že od prej. Same so začele uporabljati sporočila, ki so sicer zahtevnejša za razumevanje ravno zato, ker je prevod v slovenščino 'objavljanje sporočil' in to otroke včasih zmede, če želijo prejeti ali poslati odgovor. Dekleta so na začetku pogovor realizirala s čakanjem po nekaj sekund, vendar so s spreminjanjem kode ugotovile, da ta način ni priročen, saj se z vsako spremembo in vstavljenim blokom spremeni tudi čas izvajanja programa.
Fantje so se raje lotili druge igre. Najprej so se spoznali z neskončnim izvajanjem zanke, da so se pomaranče tako ves čas premikale in odbijale. Nato je bilo potrebno sprogramirati lopar, da sledi premikanju miške. Tudi ta potem, ko so se spoznali z ukazom 'ponavljaj', ni bil več problem. Kar veliko nerazumevanja je bilo z več pogoji v zanki, saj je bilo potrebno pri pomarančah ob dotiku loparja narediti še odboj. Mogoče bi morali narediti še kakšno bolj osnovno nalogo, da bi tudi mlajši otroci res razumeli, kako smo uporabili pogoje za dotik loparja in dotik mačka drugega za drugim v neskončni zanki.

1.6 POROČILO 6

DATUM: 10. 1. 2014

ORODJE: Programski jezik Scratch

NALOGA: Risanje krogov. Najprej v sredini narišemo en krog in okoli njega še šest enako velikih krogov s središči, enakomerno razporejenimi na prvi krožnici. Otroci so si morali pomagati z znanjem kotov, nekaj pa tudi s poizkušanjem. Naloge so bile načrtovane na podoben način, kot je risanje z želvo v Logu.

NAMEN: Učenje kotov, deljenje.

UGOTOVITVE: Nekateri otroci so hitro matematično ugotovili, kako izračunati pravo razmerje razporeditve krogov okoli sredinskega kroga. Nekaterim je pomagala gospa Irena s svojo razlago, jaz pa sem najmlajšim najprej rekla, naj malce poizkušajo, nato pa so sami ugotovili, kako izračunati, da je okoli kroga potrebnih 6 krogov z razmikom 60^o. Nekaj časa je otrokom vzelo tudi to, da so razmislili, kako končati krog okoli sredinskega, da lahko nadaljujejo z risanjem naslednjega. Starejšim otrokom je to šlo kar hitro, nekateri pa so to spet ugotovili s poizkušanjem. Mark, ki je hitro opravil z reševanjem osnovne naloge, je hitro začel risati več kot šest krogov okoli sredinskega. Poizkušal je različne kote in različno število krogov. Število krogov je shranil v spremenljivko in kar z drsnikom spreminjal število leteh. Mlajša, Marko in Gitti, sta ob risanju ugotovila, da se liki rišejo in premikajo hitreje, če spreminjamo kot za 5^o po 5 korakov naenkrat v zanki, ki se ponovi 72-krat, kot pa če se premikamo za 1^o po en korak in to 360-krat.

1.7 POROČILO 7

DATUM: 31. 1. 2014

ORODJE: Programski jezik Scratch

NALOGA: Malo drugačen ping pong. Žoga naj potuje po prostoru in se odbija od robov površine in zelene palice. Če se žoga dotakne balona, se slednji premakne za naključno število korakov v levo ali desno, žoga naj takrat spremeni smer. Če žoga pade v morje, je igre konec. Točke se prištevajo, ko se žoga dotakne rdečega balona.

NAMEN: Spremenljivke, zanke, odziv na dogodke.

UGOTOVITVE: Ker je šel krožek že proti koncu, je bilo vzdušje bolj sproščeno. Nekateri so delali naloge, ki so jih začeli že doma, spet drugi so si izmišljali svoje, nekateri so preizkušali tiste, ki so jih že objavili drugi otroci, in poskušali narediti podobne. Trije so se odločili, da bodo naredili nalogo, ki jim jo je predstavil učitelj. Enemu dekletu je naloga uspela brez pomoči, dva pa sta je nekaj potrebovala. Težave jim je delalo to, da jim samim ni uspelo sprogramirati, kako se prišteje le ena točka, ko se žoga dotakne balona. Profesor je potem povedal, da kode ne pišemo za balon, temveč za žogo, in tako je bila naloga lažja. Z odboji žoge in naključnimi koti ni bilo težav, saj so otroci te koncepte že kar nekajkrat uporabili tudi prej, tako da so to znanje že usvojili. Lea, ki je sama izdelovala svojo nalogo, je naredila program za risanje in spreminjanje barv. Vključno z vpisovanjem gesla na začetku in veliko kode v ozadju za vsako barvo, ki je lahko spremenila barvo svinčnika.

1.8 POROČILO 8

DATUM: 7. 2. 2014

ORODJE: Programski jezik Scratch

NALOGA: Spusti svinčnik in riši. Ko klikneš krogec, naj krogec pušča odtis (klon).

NAMEN: Odziv na dogodke.

UGOTOVITVE: Večina otrok je sicer reševala svoje naloge. Mark je pri reševanju svoje naloge ugotovil, da bi potreboval različne spremenljivke, kot so: int, float in double. Vendar jih ni znal tako poimenovati. Lea je nadaljevala z reševanjem svoje naloge in tokrat je poizkušala narediti barvno paleto v obliki kroga, v katerem bi na začetku nastavila določeno barvo, nato pa glede na spreminjanje kota v tem barvnem krogu zamenjala tudi barve. Alternativna rešitev je bila spreminjanje barv po Scratchevi barvni lestvici, ki jo je predlagal profesor. Mark in profesor sta imela na koncu ure pogovor, kako je lahko računalnik boljši v šahu od navadnega igralca. Predvsem je bila tema pogovora to, koliko potez vnaprej lahko računalnik preizkusi. Eno dekle je naredilo kviz z vžigalicami in animacijo le-tega.

2 Splošne ugotovitve s krožka programiranja

Na krožku so otroci večinoma imeli enako predznanje. To je bilo dobro za dinamiko skupine, saj so v začetku vsi še raziskovali Scratch in spoznavali ukaze. Le eden je že obvladal vse osnovne in tudi nekaj naprednejših funkcionalnosti Scratcha. Sam krožek ni bil začrtan kot strogo reševanje nalog, ki so bile predstavljene na začetku ure, ampak bolj kot spodbuda za samostojno programiranje in raziskovanje. Nekateri so sledili navodilom in delali tako, kot jim je bilo rečeno na začetku, drugi so te naloge še dodatno nadgrajevali, ostali pa so izdelovali čisto svoje naloge, ki so si jih bodisi izmislili sami bodisi so jih našli v Scratchevi knjižnici na spletu.

Otroke najbolj pritegne izdelovanje igric, še posebej takih, ki jih že poznajo. Poleg tega jim je veliko lažje, če so v začetku naloge krajše in razumljivejše, da koncepte sploh razumejo in jih sestavljene naloge ne zmedejo.

Opazovanje in pomoč pri krožku mi je izredno veliko pomagalo pri sestavljanju svojih nalog, saj sem zaradi tega lažje pripravila naloge za učenje programiranja, poleg tega pa mi je veliko pomagalo tudi to, da je bila zraven učiteljica razrednega pouka Irena, ki je pomagala pri razlagi matematičnih konceptov in sem se spoznala tudi s pedagoškim pristopom razlage, ki je pri pripravi nalog tudi izredno pomemben.

3 Poletna šola FRI: Čarajmo v Scratchu

Poletna šola FRI: Čarajmo v Scratchu je potekala tri dni od 30. 6. 2014 do 2. 7. 2014. Udeležilo se je je 22 otrok od 4. do 6. razreda. Njihovo predznanje je bilo različno, vendar smo s pomočjo dodatnih nalog primerno obremenili tudi starejše in tiste z več izkušnjami programiranja v Scratchu. Na treh sestankih pred poletno šolo smo začrtali okviren program, ki smo ga nato izvajali s profesorico Ireno Nančovsko, asistentko in vodjo projekta Špelo Cerar ter študentko podiplomske stopnje računalništva na Pedagoški fakulteti Barbaro Stopar.

3.1 POROČILO 1. dan

DATUM: Ponedeljek, 30. 6. 2014

POROČILO: Prvi dan smo se na delavnicah predstavili preko predstavitve, pripravljene v Scratchu. Najprej smo ustvarili uporabniške račune za vse udeležence, ki smo si jih skupaj z gesli zapisali, da jih niso pozabili. Nato smo mentorji pokazali svoje predstavitve. Preden so se začeli ukvarjati z iskanjem blokov in prvih ukazov, smo še skupaj ustvarili animacijo, kako se premika rak. Predstavila sem jim bloke za premikanje, povečevanje, skrivanje z dodajanjem figur in spreminjanjem ozadja. Otroci so z lahkoto sledili in ustvarili program po navodilih. S tem so pridobili dovolj znanja, da so nadaljevali z ustvarjanjem svojih predstavitev. Nekateri so imeli že več predznanja in so tako uporabili več ukazov, ki so bili tudi zahtevnejši. Nezavedno so začeli uporabljati zaporedno izvajanje ukazov. Uporabljali so predvsem bloke za gibanje in spreminjanje videza. Nekaj jih je s pomočjo mentorjev uporabilo tudi bloke za sporočila. Za naslednjič bi lahko izboljšali izvajanje tega dela programa tako, da bi otrokom dali nekaj opornih točk, ki jih morajo nujno predstaviti, da se dolžina predstavitev ne bi toliko razlikovala (primer: v predstavitvi naj povejo vsaj, kateri razred so zaključili, odkod so doma in koliko imajo bratov in sester.) Opaznih težav z neznanjem uporabe računalnika in težav z osnovno uporabo Scratcha ni bilo, saj so vsi ustvarili kvalitetne predstavitve.

Pred kosilom je sledila še predstavitev ustvarjenih programov, ki smo ji vsi pozorno prisluhnili. To je dobra praksa, saj tako otroci vidijo še več idej, ki so jih uporabili njihovi kolegi, in tako razmišljajo, na kašen način so se nalog in programiranja lotili drugi. Poleg tega je za otroke nekakšen izziv, da se pokažejo pred prijatelji in predstavitev naredijo bolj kvalitetno in kot zaključeno celoto ter tako ne ostane nedokončana.

Po kosilu smo imeli igro Risanje po navodilih, ki je sicer naloga z vzporedne Poletne šole FRI: Računalništvo brez računalnika. Tu so otroci spoznali, kako pomembno je dajati natančna navodila računalniku, da lahko izvede določen program.

Kasneje smo ustvarili igro Labirint. Mentorica Špela je otroke počasi vodila skozi ustvarjanje programa. Z njeno pomočjo so po ukazih skupaj izdelali program, ki so ga na koncu lahko še sami nadgradili. Najprej je bilo potrebno pripraviti ozadje in ga narisati. Nato so sprogramirali premikanje miši s pomočjo pritisnjenih tipk, kar otrokom ni delalo večjih težav. Težje jim je bilo v začetku razumeti, kaj se zgodi, če se miš dotakne stene. Program smo namreč naredili tako, da se premakne za toliko korakov nazaj, kot se navadno premakne naprej. Po nekaj poizkušanjih je bilo vsem jasno, zakaj je tako. Za tem smo dodali še kolaček, do katerega je morala miš priti skozi labirint. Ko se je miš dotaknila kolačka, je ta spremenil obliko v obgrizenjen kolaček. S postavljanjem teh pogojev otroci niso imeli bistvenih težav. V naslednjih dneh so labirint še izboljšali, saj so ga morali predstaviti na zaključni predstavitvi ostalim otrokom in mentoricam. Nastale so zanimive verzije labirintov z dodanimi ovirami, kot so mačke, bombe, strele. Nekateri so naredili več ravni, tako da se ob dotiku kolačka spremeni labirint, miš pa se postavi na začetno točko. Neki fant je dodal točke, in sicer tako, da je ob dotiku miši ob steno labirinta vsakič izgubil eno točko. Nekateri so se znašli tako, da so miš naredili hitrejšo od mačke. Tako smo imeli na koncu res veliko različnih verzij labirintov z različnimi dodatki.

Tekom dneva smo imeli veliko težav s tem, da figure niso bile centrirane, saj je prihajalo do težav z nepravilnim obračanjem figur ali zaznavanjem dotikov.

Prvi dan je bil uspešen, naredili smo vse, kar smo načrtovali.

3.2 POROČILO 2. dan

DATUM: Torek, 1. 7. 2014

POROČILO: V torek dopoldne smo izdelovali igro ping pong. Tudi ta dan smo se izdelave programa lotili skupaj po navodilih in s sodelovanjem učencev. Najprej smo na tablo zapisali, kakšno je okolje pri igri ping pong, katere figure potrebujemo, kako bomo realizirali odboje in kako izgubo žogice. Najprej smo ustvarili ozadje, kjer sta bili na obeh robovih različni barvi in ob dotiku žogice s to barvo je nasprotnik dobil točko. Po pripravi ozadja smo pripravili žogico, ki se je v začetku odbijala le od roba zaslona. Kasneje smo po pripravi loparjev, ki so se premikali le gor in dol, dodali še odboj od loparjev. Igra je bila narejena za 2 igralca. Nekateri so jo naredili tudi tako, da je igrala proti računalniku. Dobro je bilo to, da otroci te igre še niso sami sprogramirali, saj bi jih mogoče kar malo prestrašilo, da bi morali sami pripraviti tako zahtevno igro. So pa popoldne sprogramirali podobne igre in so tako sami napisali program, ki se odziva na pritisnjene tipke s tipkovnice in podobne dogodke kot pri ping pongu.

Nato smo pripravili še štetje točk z uporabo spremenljivk, ki so jih otroci z lahkoto doumeli. Za boljše smo predlagali, da dodajo še zvočne učinke in

naj žogica namesto na sredini polja začne svojo pot pri tistemu, ki mu tudi po pravilih ping ponga pripada servis. Prepričana sem bila, da otrokom ne bo lahko ugotoviti, kdo je zadnji prejel točke, vendar sem se uštela in bila presenečena, kako hitro so popravili program tako, kot sem želela.

Po kosilu so imeli bolj proste roke pri tem, kaj bodo izdelali. Predlagali smo izdelavo zgodbice, ki temelji na kakšni resnični pravljici ali zgodbi. Drugi predlog je bil, da sami pripravijo dirkalno stezo in avtomobil, ki se bo odzival na pritisnjene tipke levo, desno in presledek, ki bi pomenil pospešek. Fantje so seveda takoj začeli z izdelavo slednje igre. Tudi dekleta niso bila tako navdušena nad pripovedovanjem zgodb, ampak so v večini sprogramirale igrice, kjer z neba padajo različni predmeti, figura na spodnjem robu ekrana pa jih mora loviti. Nekatere so dodale tudi nevarne predmete, ki so odštele pridobljene točke. Težava, ki se je pojavila pri programiranju teh predmetov, je bila, da do šestega razreda večinoma še ne poznajo koordinatnega sistema in je bilo potrebno kar podrobno predstaviti, kaj pomenita koordinati x in y.

Otroci so to popoldne aktivno programirali, saj smo jih motivirali tako, da smo jim rekli, da bodo naslednji dan predstavili svojo igro labirint in še eno igro ali zgodbo, ki jo bodo sprogramirali.

3.3 POROČILO 3. dan

DATUM: Sreda, 2. 7. 2014

POROČILO: Za zadnji dan smo pripravili program samo še dopoldne, popoldne pa smo imeli predstavitve nalog, ki so jih otroci ustvarili v času poletne šole.

Predstavila sem jim nalogo z večkotniki, ob tem pa so spoznali tudi vnašanje besedila in uporabo spremenljivk, ki jih je nekaj udeležencev uporabljalo že dan pred tem pri predstavitvi pripovedovanja zgodb v Scratchu. Program smo napisali po korakih. Najprej smo narisali kvadrat tako, da smo narisali črto in se obrnili ter ta ukaz ponovili štirikrat. To je s premikanjem po tleh prikazal kar eden izmed udeležencev. Nato smo na tablo zapisali, kako smo s koti prišli do tega: 4 x 90 = 360. Nato smo na ta način zapisali še formulo za trikotnik: 3 x 120 = 360. Otrokom se je hitro posvetilo, da na ta način lahko izračunamo kot za vse n-kotnike. Namesto ročnega vstavljanja kotov smo nato vstavili spremenljivko odgovor, ki smo ji z vsakim odgovorom uporabnika priredili vrednost.

Naslednji del dopoldneva so otroci dokončali svoje igre za predstavitev, ki so jih ostalim pokazali po kosilu. Sledila sta še podelitev diplom in slikanje.

4 Splošne ugotovitve s Poletne šole FRI: Čarajmo v Scratchu

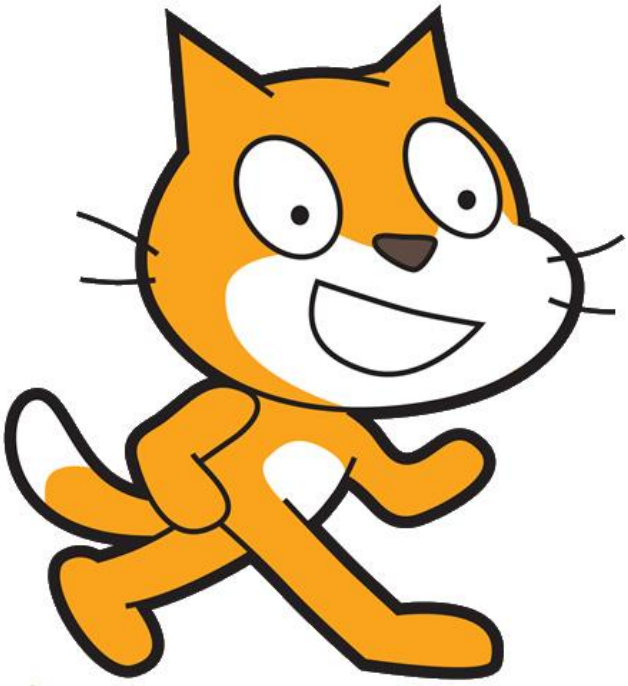
Večdnevno aktivno učenje programiranja je veliko lažje kot enotedensko po eno uro. Otroci lahko izdelujejo veliko kompleksnejše igre. Dobro je bilo tudi, da je bil na vsake štiri otroke po en mentor, kar je pomagalo pri tem, da je vsak otrok ob nerazumevanju dobil pomoč. Pri takšni starostni skupini je še posebej pomembna motivacija, kot je na primer predstavitev projekta pred drugimi, ker drugače otroci hitro zaidejo na druge spletne strani ali pa preizkušajo igre drugih v Scratchevi knjižnici. Ta oblika poučevanja programiranja je zelo dobra, vendar bi bilo za nadaljevanje tega projekta potrebno poskrbeti še za več nadgradenj za hitrejše in tiste, že vešče v Scratchu.
Programsko je bila Poletna šola dobro začrtana, saj smo otroke strukturirano naučili programiranja in programskih konceptov, le da tega nismo tako zelo poudarjali.

Dodatek B

Naloge za krožek računalništva v programskem jeziku Scratch

NALOGE ZA KROŽEK RAČUNALNIŠTVA

v programskem jeziku SCRATCH



KAZALO

UVOD

Knjižica je nastala kot priloga k diplomski nalogi in je namenjena učiteljem, mentorjem, staršem ali starejšim bratom in sestram, ki želijo otroke učiti programirati. Njihovo predznanje ni pomembno, saj so naloge razložene tako, da jih lahko razume in naredi vsak.

Danes se otroci zelo zgodaj srečajo z računalnikom. Nekatere hitro premamijo igre, spet drugi se skozi igro učijo. Veliko dobrih programerjev pravi, da so se s programiranjem začeli ukvarjati že v osnovni šoli.
Otroci oziroma mladostniki v tem obdobju še neobremenjeno razmišljajo tudi o zahtevnih programskih strukturah in elementih ter se včasih z njimi spopadajo na drugačen način kot starejši programerji in tisti, ki začnejo programirati v poznejših letih.

Komu je priročnik namenjen?
S knjigo, ki je namenjena učiteljem in tistim, ki bodo otroke učili osnov programiranja, želimo otrokom približati svet logičnega razmišljanja. Tudi če učitelj oziroma pedagog nima nobenih izkušenj na področju programiranja ali računalništva, bo s pomočjo teh navodil z lahkoto pridobil vsa zahtevana znanja, da otroke vodi in spodbuja na pravi poti programerskega razmišljanja.

Zakaj smo izbrali ravno Scratch in ne katerega drugega orodja oziroma programskega jezika, namenjenega otrokom?
Najprej predvsem zato, ker je edini na voljo v slovenskem jeziku in tako otroci nimajo težav z razumevanjem ukazov.
Prednost programiranja v Scratchu za otroke je tudi to, da se jim ni potrebno ukvarjati s sintaktičnimi napakami, saj so bloki z ukazi že ustvarjeni, oni jih le povlečejo v glavno okno in spreminjajo le del ukazov. Tako se lahko še bolj posvetijo programiranju in razmišljanju o sestavi programa in ne le uporabi ukazov, s katerimi imajo programerji − začetniki nemalo težav.

Programski koncepti, ki se jih bomo naučili preko reševanja sestavljenih nalog:

- zaporedje izvajanja ukazov,
- zanke,
- pogojni stavki,
- spremenljivke,
- tabele,
- odziv na dogodke,
- niti,
- časovna usklajenost in sinhronizacija;
- vnos podatkov preko tipkovnice,
- naključna števila,
- logične operacije,
- dinamična interakcija.

Pri reševanju nalog, ki vas čakajo, ni možna le ena rešitev, ampak jih je neskončno. Na koncu je važno, da program deluje pravilno, kot so zahtevala navodila. Seveda so nekatere rešitve bolj pravilne od drugih, vendar bo otrok do tega, katera je boljša, prišel skozi čim več sprogramiranih nalog.

Naloge so krajše in ne zahtevajo več ur dela. Tako je možno narediti vsako 45-minutno uro vsaj eno.

Razdelitev knjige:
V priročniku smo se najprej osredotočili na predstavitev okolja Scratch, nato pa sledijo naloge z navodili in pomočjo, kako naj učitelj naloge predstavi učencem in jih vodi skozi reševanje do prave rešitve.

Pri prvih nalogah je predlagano, da jih rešite frontalno, skupaj z učenci. Kasneje sledite navodilom po korakih in jih z njihovo pomočjo usmerjajte, če boste opazili, da otroci sami ne znajo sprogramirati naloge. Predvideno je, da se najprej pokaže končni izdelek, na povezavi z materiali in ne programske kode, da si otroci predstavljajo, kaj morajo narediti. Primer ogleda končnega izdelka brez kode je na povezavi: http://screencast.com/t/A7xfgTUDGY3. Nato so koraki za vsako nalogo posebej opisani.

Želim vam veliko zabave pri programiranju,
Petra

UPORABA SCRATCHA

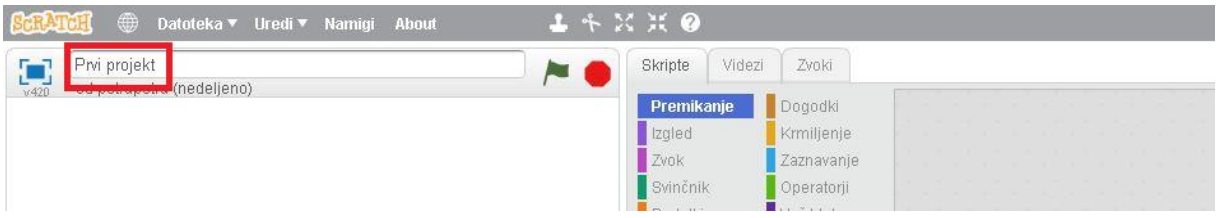
Za izvajanje krožka potrebujemo računalnik za vsakega udeleženca in internetno povezavo. Obstaja sicer tudi aplikacija, ki omogoča programiranje brez internetne povezave, vendar v tem trenutku še ni na voljo v slovenskem jeziku in ni priporočljiva, ker želimo otrokom omogočiti, da svoje delo na teh učnih urah shranijo in urejajo tudi doma.

Najprej se prijavimo na spletni strani: http://scratch.mit.edu/
Če naš privzeti jezik še ni slovenski, ga nastavimo čisto na koncu strani, v desnem robu:

V zgornjem desnem kotu kliknemo »Pridruži se Scratchu«, kjer ustvarimo nov račun:

Pri vpisu le sledite navodilom. Predlagamo vam le, da si zapomnite tudi otrokovo uporabniško ime in geslo, saj ju ta hitro pozabi.
Sedaj lahko začnete programirati! Nov projekt ustvarite tako, da v levem zgornjem kotu kliknete: »Ustvari.«

Projekt poimenujemo v polju, ki se nahaja na zgornji levi strani ekrana:



ZAPOREDJE UKAZOV

Kaj pomeni zaporedje ukazov?

Navodila za sestavljanje igračke v čokoladnem jajčku, recept za najboljše piškote ali palačinke ...

Vse to so programi, katerih izvajanje poteka v nekem zaporedju. Navadno ne moremo namreč spremeniti zaporedja priprave palačink tako, da najprej spečemo jajca in jih šele potem razbijemo. Tudi programerji napišemo program tako, da si ukazi sledijo v zaporedju in jih računalnik v takem vrstnem redu tudi izvaja.

Vesoljec na luni

Naloga

Vesoljec se sprehaja po luni. Najprej naj se premakne za nekaj korakov naprej. Nato naj nas pozdravi in reče: "Pozdrav z lune!" Potem naj nadaljuje svojo pot tako, da se ponovno sprehodi nekaj korakov naprej.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- se prijaviti v Scratch, poimenovati in shraniti projekt.

Učenci se v okviru naloge naučijo:

- dodati ozadje,
- dodati novo figuro,
- da je za zagon programa potrebno dodati pogoj: »če pritisnem zastavico, izvedi«,
- zaporednega izvajanje ukazov,
- iskati osnovnih ukazov za premikanje in pogovor.

Materiali

http://scratch.mit.edu/projects/15396791/

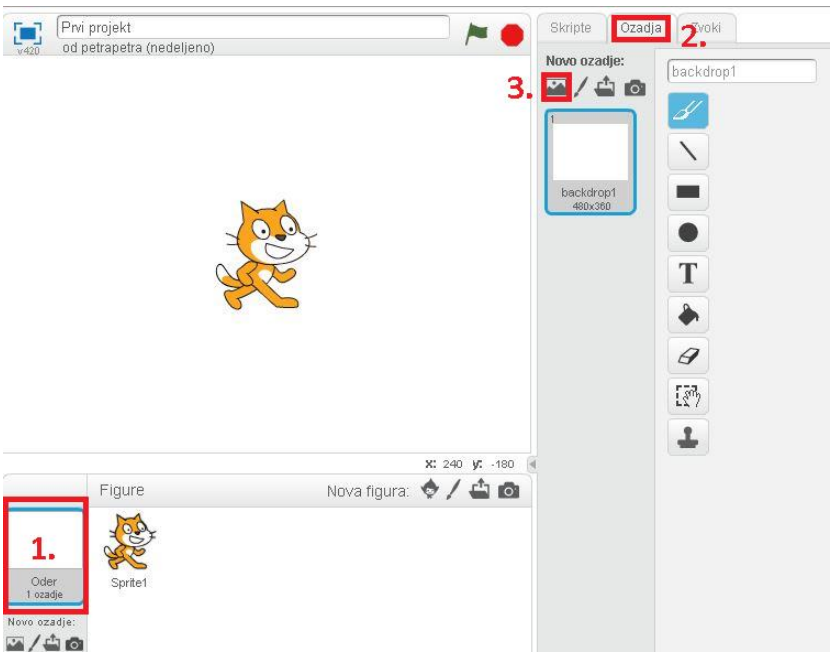
Potek

Učencem pokažemo končni izdelek. Pokažemo, da se začne program izvajati, ko pritisnemo zeleno zastavico. Program se nato izvede do konca.

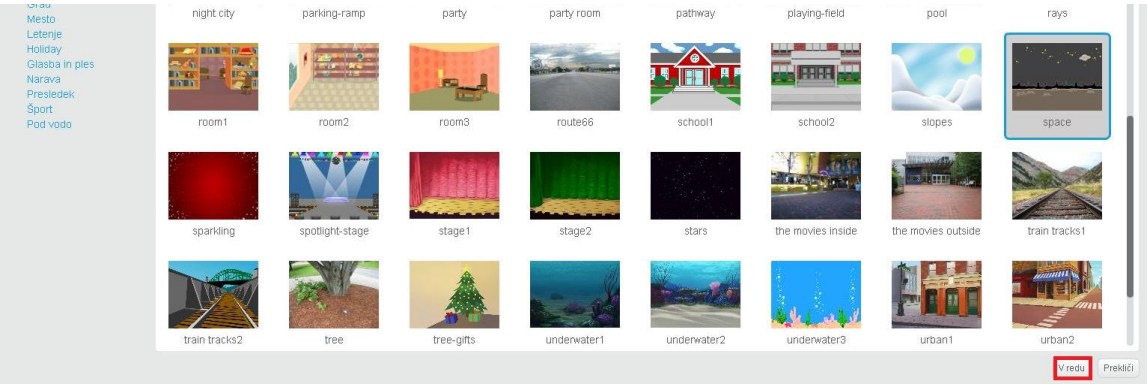
1. Priprava grafične podobe

   Najprej naj otroci zamenjajo ozadje, ki naj bo čim bolj primerno temi. Za tem naj spremenijo še vizualni izgled figure. Predlagam, da nalogo rešujete frontalno, otroci naj vam sledijo in rešujejo nalogo na svojem računalniku.

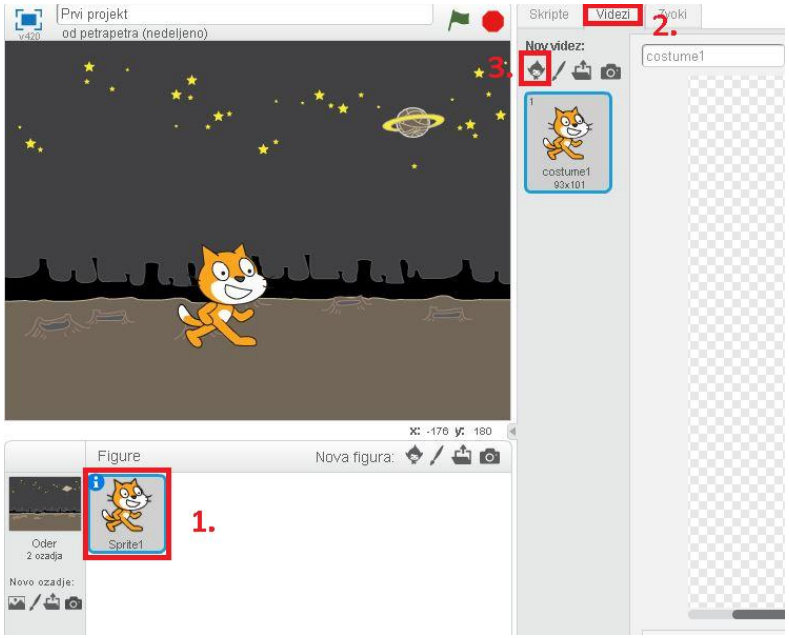
   Menjava ozadja po korakih:



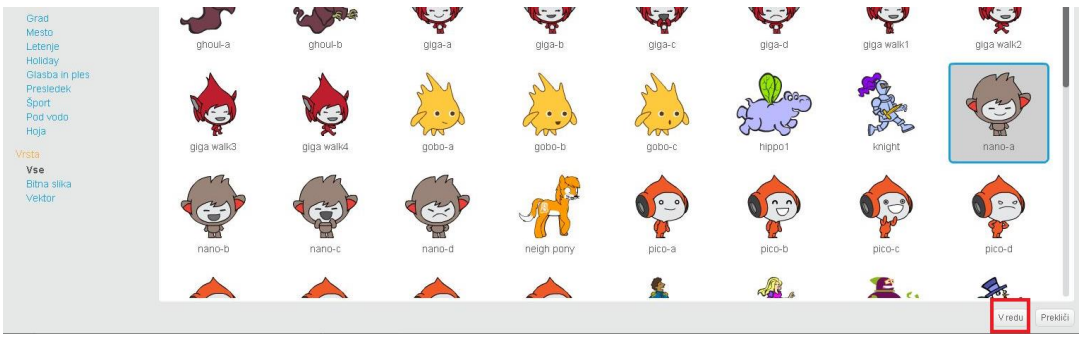
Izberemo enega izmed ozadij in kliknemo »V redu«



Spreminjanje vizualnega izgleda figure poteka podobno:



Izberemo ustrezno figuro in kliknemo »V redu«



2. Premikanje

Nato naj med podanimi elementi izberejo prave bloke za premik naprej. Dodajo naj ukaz, ki bo ta premik sprožil ob kliku na zastavico.

Opozorilo: Aktivni zavihek mora biti tisti z imenom »Skripte,« ker lahko šele takrat povlečemo blok z ustrezno kodo na polje za pisanje programa:

Dodatne možnosti

Vesoljec se najprej premakne 50 korakov, reče: "Pozdrav iz lune!", se ponovno sprehodi 50 korakov. Nato naj pomisli: »Sedaj pa bom izginil!« in naj izgine.

Rešitev

Lep dan v morju

Naloga

V morju je lep dan. Morska zvezda naj malce zaplava po morju in reče: »Kako je lep dan!«. Pogleda naj še v nebo in povei: »Pa še sonce sije!« Nato naj zaplava v smer, v katero je gledala že na začetku.

Predznanje in cilji

Predvidevamo, da so učenci že usvojili naslednja znanja:

- ustvarjanje novih projektov,
- menjanje ozadja,
- uporabe osnovnih blokov za premikanje naravnost,
- iskanje bloka za govorjenje.

Če se učenci v sklopu osnovnega šolanja še niso srečali s koti in koordinatnim sistemom, je sedaj pravi čas, da jih spoznate/seznanite z osnovami. V tej nalogi bo zadostovalo, če jim razložite le kote.

Učenci v okviru naloge:

- sami iščejo bloke, ki jih potrebujejo za reševanje naloge, in ob tem raziskujejo, kakšni bloki vse obstajajo,
- se srečajo s koti in njihovo uporabo,
- se naučijo, kaj pomeni zaporedno izvajanje ukazov.

Materiali

http://scratch.mit.edu/projects/15016093/

Potek

Tudi to nalogo rešite skupaj z otroki, oni pa naj vam sledijo in obenem programirajo tudi svojo nalogo.

1. Pokažemo končni izdelek
   Učencem pokažemo animacijo izdelka na zgornji povezavi, ne pokažemo pa jim rešitve naloge.

2. Priprava ozadja in izbira figure
   Izberemo ozadje in figuro glede na tematiko naloge.

3. Premik naprej
   Premik naprej smo se naučili sprogramirati že v prejšnji nalogi.

4. Zvezda naj reče: »Kako lep dan!«
   Pri tem ukazu je potrebno paziti, da se besedilo obdrži na zaslonu toliko časa, da ga je možno prebrati. Otroci naj sami poiščejo primeren blok.

5. Zvezdica naj pogleda proti soncu
   To stori tako, da se obrne za, na primer, 30 stopinj v levo.

6. Zvezdica naj reče: »Pa še sonce sije!«
   Ukaz izvedemo na enak način kot v točki 4.

7. Vrnitev v lego, kamor je element gledal na začetku
   Obrne naj se za 30 stopinj v desno oziroma za toliko, kolikor se je prej premaknila v levo. Kote naj otroci poljubno spreminjajo, da se navadijo njihove uporabe.

Dodatne možnosti

Po tem, ko zvezda reče: "Kako lep dan!", poskoči.
Ko zvezdica skoči, naj v zraku počaka kakšno sekundo, da opazimo, da je skočila.

Rešitev


ko je kliknjena
pojdi 50 korakov
povej Kako lep dan! za 2 sekund
pojdi 50 korakov
obrni se za 30 stopinj
povej Pa še sonce sije! za 2 sekund
turn 30 degrees


ko je kliknjena
pojdi 50 korakov
povej Kako lep dan! za 2 sekund
pojdi 50 korakov
spremeni y za 50
počakaj 1 sekund
spremeni y za -50
počakaj 1 sekund
obrni se za 30 stopinj
povej Pa še sonce sije! za 2 sekund
turn 30 degrees

Balerina

Naloga

Balerino najprej postavi na koordinate x = -160, y = -30.

Nato naj drsi v koordinatno izhodišče x = 0, y = 0.

Balerina naj nato poskoči. Za tem naj se prikloni tako, da se nagne v desno za 15 stopinj in se vrne nazaj v navpičen položaj.

Na koncu naj zapusti oder tako, da se premakne naravnost pod oder.

Predznanje in cilji:

Predpostavljamo, da učenci že poznajo:

- zaporedno izvajanje ukazov, iskanje blokov, spreminjanje ozadja in figur,
- osnove koordinatnega sistema, če jih ne, je potrebno razložiti osnove:
  - kaj pomeni izhodišče koordinatnega sistema, ta je pri Scratchu v središču slike,
  - razložiti je potrebno tudi, kaj pomeni pozitivni del osi, kaj negativni. Poleg tega obrazložimo, da se za premik v desno spremeni le x-koordinata, za premik v levo x-koordinata, ki mora biti negativno predznačena, za premik navzgor se prišteje vrednost y-koordinati, za navzdol pa se y-koordinati odšteje želena vrednost,
  - kaj se zgodi, če spremenimo obe koordinati,

- osnove kotov:
    - kako se figura nagne v desno in levo s pomočjo kotov.

Učenci v okviru naloge:

- utrdijo znanje koordinatnega sistema, mlajši učenci ga šele spoznajo,
- naučijo ali utrdijo znanje premikanja figur po koordinatnem sistemu, saj jim bo to prišlo prav pri veliko nalogah iz Scratcha,
- naučijo se kotov ali utrdijo znanje le-teh – premikanje in sukanje figur.

Materiali: http://scratch.mit.edu/projects/15515679/

Potek: Glede na predznanje učence naučimo uporabljati koordinatni sistem in kote, tkot je opisano zgoraj. Če učenci to že razumejo, znanje le obnovimo.

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure
   Izberemo ozadje in figuro glede na tematiko naloge.

3. Postavitev figure na določeno mesto
   Figuro v Scratchu postavimo na pravo mesto tako, da podamo x- in y-koordinato.

4. Premakni se v koordinatno izhodišče
   Premik v koordinatno izhodišče, tako da spremenimo x- in y-koordinato na 0. Uporabimo tak ukaz, da balerina drsi do tega mesta.

5. Obdrži sliko na zaslonu
   Ker se ukazi izvajajo tako hitro, v naslednjih korakih ne bomo mogli videti, kaj se dogaja med začetnim in končnim stanjem. Zato je potrebno med posamezne ukaze za premike dodati še en ukaz, da figura vmes počaka kakšno sekundo.

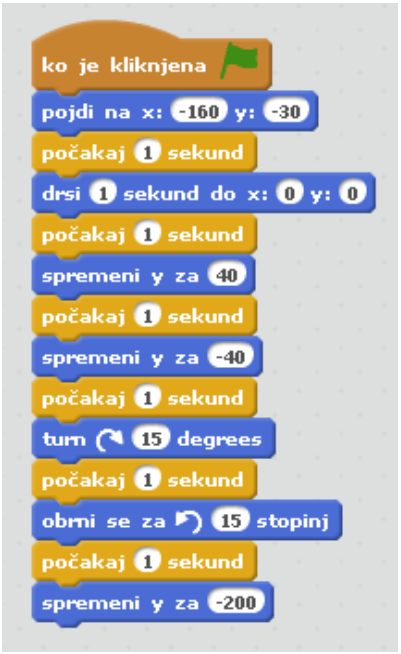
6. Figura naj poskoči
   Da figura poskoči, je potrebno spremeniti y-koordinato v pozitivno smer, malce počakati, ter spremeniti y v negativno smer za toliko, kolikor smo ga prej spremenili v pozitivno.

7. Figura naj se nagne v desno
   Za nagib v desno uporabimo ukaz za spremembo kota za nekaj stopinj, počakamo z ukazom za čakanje in se vrnemo za tolikšen kot, kot smo spremenili pred tem.

8. Figura naj se nato premakne na dno scene
   Figuro premaknemo s spreminjanjem y-koordinate.

Dodatne možnosti: Balerina naj pred priklonom naredi še obrat, tako da spreminja kote za 90 stopinj z ukazom: ”obrni se v smer”.

Rešitev:



ZANKE

Naloge z učenjem zank so napisane tako, da lahko v eni šolski uri predelate več kot eno.

Kaj so zanke?

Zgodi se, da je kdaj kakšen stavek potrebno napisati večkrat. Če ga napišemo enkrat, ni problem, če ga dvakrat, nas že malo boli roka, če pa ga je potrebno napisati stokrat, je to že večji problem. Kaj šele, če ga moramo napisati tolikokrat, dokler nam ne zmanjka listov?

Nekajkratno ponavljanje ukaza ali sklopa ukazov imenujemo zanka.

Včasih vemo, kolikokrat hočemo izvršiti nek ukaz in lahko napišemo, da želimo stavek napisati trikrat.
Včasih pa ne vemo točne številke, ampak vemo, da nehamo pisati, ko zazvoni šolski zvonec. Vse to so zanke.

Izgubljena žoga

Naloga

Nekdo je na igrišču pozabil žogo. Zapiha veter in žoga se začne kotaliti po igrišču sem ter tja ter celo malo poskakuje.

Napiši program tako, da se žoga ves čas premika, če se dotakne roba prizorišča, pa naj se odbije.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati osnovne bloke in poiskati nove,
- uporabljati kote.

Učenci v okviru naloge:

- spoznajo uporabo zank,
- razumejo, kaj pomeni neskončna zanka – da se program v zanki neprestano izvaja.

Materiali

http://scratch.mit.edu/projects/15518577/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure
   Izberemo ozadje in figuro glede na tematiko naloge.

3. Uporaba neskončne zanke
   Žoga se bo kotalila od trenutka, ko bomo pritisnili na zeleno zastavico, dokler ne pritisnemo rdečega znaka za zaustavitev naloge. To pomeni ves čas. Poiskati moramo tak blok, ki bo ves čas ponavljal ukaze, ki bodo v njem napisani. Namig, če ga otroci ne najdejo: nahaja se v sekciji z bloki za krmiljenje (»ponavljaj«). Če uporabimo zanko »ponavljaj«, pomeni, da se dogodki, ki so zapisani v tej zanki, neprestano zaporedno izvajajo. Izvajanje se konča, ko pritisnemo rdeč znak stop poleg zelene zastavice v desnem kotu prikaznega polja.

4. Žoga naj se premika
   V zanko »ponavljaj« tako potem vstavimo ukaze za premikanje (npr.: »pojdi 10 korakov«).

5. Žoga naj se odbije od roba prikazne površine
   Če program zaženemo sedaj, se nam lahko zgodi, da žoga pobegne iz vidnega polja. Tako za ukazom za premikanje dodamo še ukaz, odbij se, če si na robu, in žoga se bo sedaj odbijala in ne bo ušla iz vidne površine.

6. Žoga naj se ne odbija le levo in desno
   Problem nastane, če se žoga začne premikati le levo in desno. Zato je potrebno žogo pred

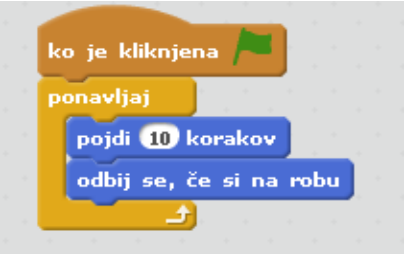
prvim izvajanjem malce zasukati, da se le-ta začne odbijati pod nekim kotom. To naredimo le pred prvim izvajanjem in tega niti ne dodamo v kodo. Ko je figura žoge izbrana, le dvakrat kliknemo ukaz za spreminjanje kotov. Kot naj bo na primer velik: 15º.

Dodatne možnosti

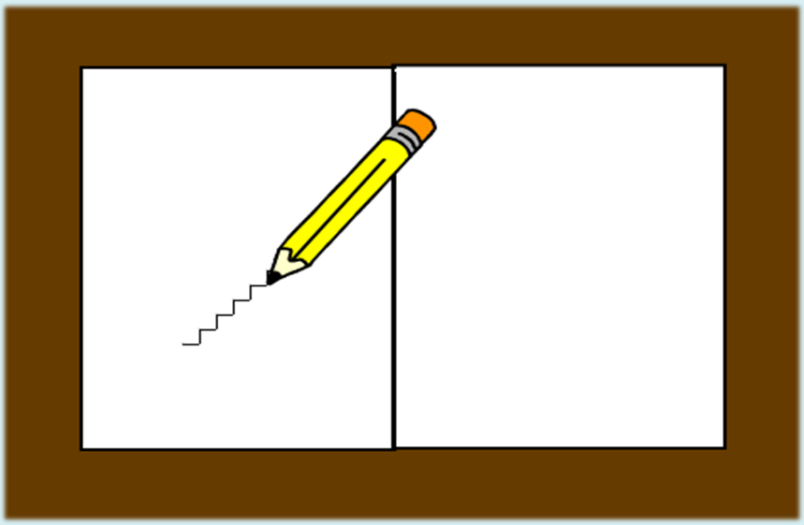
Otroci naj razmislijo, kako bi pohitrili premikanje žoge in kako bi jo upočasnili.

Kot dodatno nalogo naj učenci namesto, da se žoga vrti in odbija v vse smeri, nalogo izdelajo tako, da se žoga odbija od tal.

Rešitev



Risanje stopnic



Naloga

Za ozadje si pripravi zvezek. Vanj nato s svinčnikom nariši 10 stopnic.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati osnovne bloke in poiskati nove,
- osnove koordinatnega sistema in kotne funkcije.

Učenci se v okviru naloge:

- naučijo spreminjati in risati ozadje tako, da ga sami narišejo in popravijo,
- spoznajo z novim programskim konstruktom: zankami.

Materiali

http://scratch.mit.edu/projects/15519258/

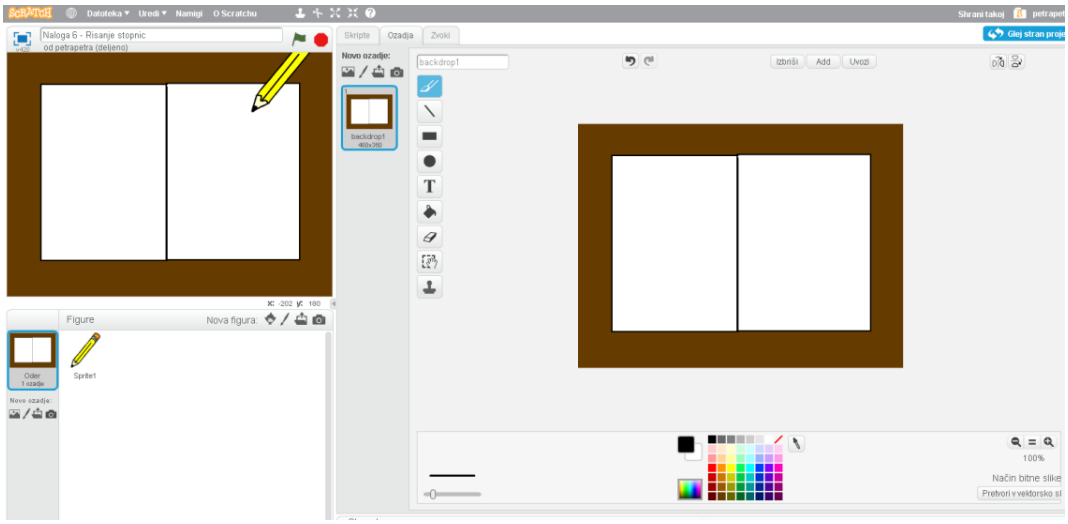
Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure

   Glede na tematiko naloge izberemo figuro, ozadje pa morajo otroci sami narisati. Narisali naj bi zvezek, v katerega bomo potem s figuro svinčnik »pisali«.

   Z miško moramo klikniti na ikono »Ozadje«, ki se nahaja na levi strani pod prikazno površino. Potem izberemo zavihek Ozadje še na zgornjem delu površine, kamor navadno pišemo kodo. Nato s pomočjo risalnih funkcij narišemo odprt zvezek.



3. Risanje ene stopnice

   Najprej s pomočjo funkcij za risanje s svinčnikom narišemo eno stopnico. Najprej bo potrebno spustiti svinčnik, se premakniti naprej za približno 10 korakov, potem pa zasukati za nek kot in ponovno za enako število korakov premakniti naprej in ponovno zasukati. Naj otroci sami poskušajo ugotoviti, za kakšen kot (90º) se je potrebno zasukati, da bomo narisali eno stopnico. V pomoč pri reševanju te naloge je tudi to, da si otroci stopnico narišejo in ugotovijo, v katero smer naj se obrnejo.

4. Risanje desetih stopnic

   Ukaze bi lahko skopirali in jih zložili enega za drugim in tako narisali 10 stopnic. Vendar s tem napišemo veliko nepotrebnih ukazov, program pa je dolg in velikokrat nepregleden. Lažje je, če uporabimo zanko, s katero risanje stopnice večkrat ponovimo.

Za ukaz "spusti svinčnik" ni potrebno, da je v zanki. Naj razmislijo, zakaj je tako. (Odgovor: Ker nam ni potrebno vsakič, ko je svinčnik že spuščen, ponovno spustiti svinčnika. S tem se znebimo odvečnega izvajanja ukazov. Predlagam, da se o tem pogovorite, ko program napišete vsi, in naj otroci programe/rešitve med sabo primerjajo.)

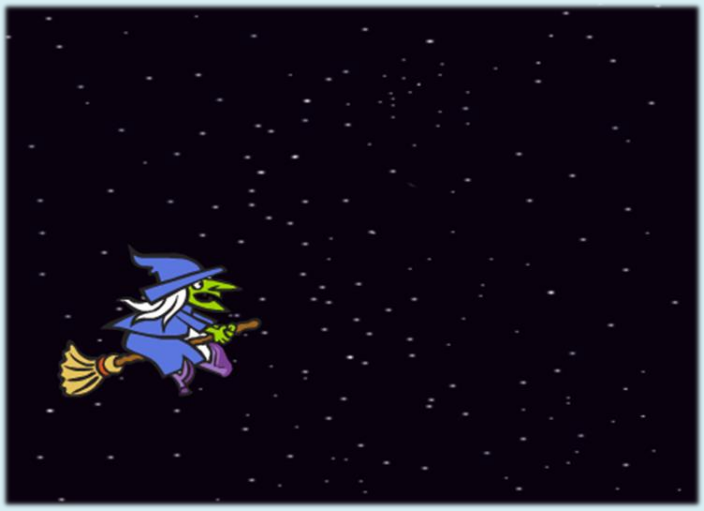
S premislekom se lahko znebimo kar nekaj vrstic nepotrebne kode. Tako smo že z zanko za vsaj 36 vrstic skrajšali program. S tem navadno naredimo program bolj čitljiv in pregleden, še posebej je to dobro, če napisano kodo bere kdo drug.

Dodatne možnosti

Ko narišeš deset stopnic navzgor, jih nariši še deset navzdol.

Rešitev

Čarovniški let



Naloga

Čarovnica naj se ves čas premika proti kazalcu miške. Ko se dotakne roba, naj se igra konča.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati bloke in poiskati nove,
- osnove zank.

Učenci v okviru naloge:

- ponovijo svoje znanje uporabe zank in se naučijo postavljati pogoje za končanje zanke.

Materiali

http://scratch.mit.edu/projects/15519827/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure
   Izberemo ustrezno ozadje in figuro.

3. Čarovnica naj se ves čas premika proti kazalcu miške
   Najprej program realiziramo tako, kot že znamo iz naloge, kjer je poskakovala žoga. Uporabimo neskončno zanko, vanjo damo ukaze za premikanje in ukaz, da se čarovnica ves čas obrača proti kazalcu miške.

4. Čarovnica naj se premika proti kazalcu miške, dokler se ne dotakne roba
   Če se čarovnica dotakne roba, se program ustavi. To pomeni, da se ves čas, ko se čarovnica ne dotika roba, premika po površini. Potrebno bo zamenjati ukaz ponavljaj. Namig, če učenci sami ne ugotovijo ali ne najdejo ukaza: uporabijo naj “ponavljaj dokler” “se dotika x”.

Dodatne možnosti

Namesto, da se čarovnica neha premikati, ko se dotakne roba, naj se ustavi, če se dotakne drevesa ali kakšne druge figure. To pomeni, da bo potrebno dodati še eno figuro in spremeniti ukaz ponavljaj. Sedaj moramo sprogramirati nalogo tako, da ponavljamo, dokler se čarovnica ne dotakne druge figure. Kodo moramo spremeniti, ko je izbrana figura čarovnica in ne druga figura.

Rešitev

POGOJNI STAVKI

Kaj so pogojni stavki?

Oče reče: "Če boš priden, greš lahko na igrišče igrat nogomet, če ne, boš ostal notri!"
Oče je pri tem, ne da bi vedel, uporabil pogojni stavek. Oče programer je tako sestavil osnovni pogojni stavek.
Če bo otrok priden, bo lahko odšel ven, če pa ta pogoj ne bo izpolnjen, bo ostal notri.
Oče bi lahko dodal še več pogojev: "Če pri testu dobiš več kot 3, dobiš sladoled, če dobiš oceno 3, ne dobiš ničesar, če pa manj, se boš moral še veliko učiti!"
Vsak dan tako srečamo v našem življenju ogromno pogojnih stavkov. Se spomniš še kakšnega?

Nogometni vratar

Naloga

Za ozadje nariši gol. Pred gol postavi fanta, ki brani. Če se ga žoga dotakne, naj reče: "Branil sem gol!", če pa se ga žoga ne dotakne, naj reče: "Ni mi uspelo."

Ker zaenkrat še ne znamo dovolj, je potrebno žogo najprej premakniti na neko mesto in nato klikniti zastavico za začetek.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati že znane bloke in poiskati nove.

Učenci v okviru naloge:

- se učijo pogojnih stavkov,
- delajo z več figurami,
- spoznavajo osnove objektnega programiranja: programi pripadajo posameznim figuram (objektom).

Materiali

http://scratch.mit.edu/projects/15516576/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure

   Ozadje narišejo otroci sami s pomočjo urejevalnika v Scratchu ali ga uvozijo iz računalnika. Dodajo dve različni figuri: golmana in žogo.

3. Uporabimo pogojni stavek

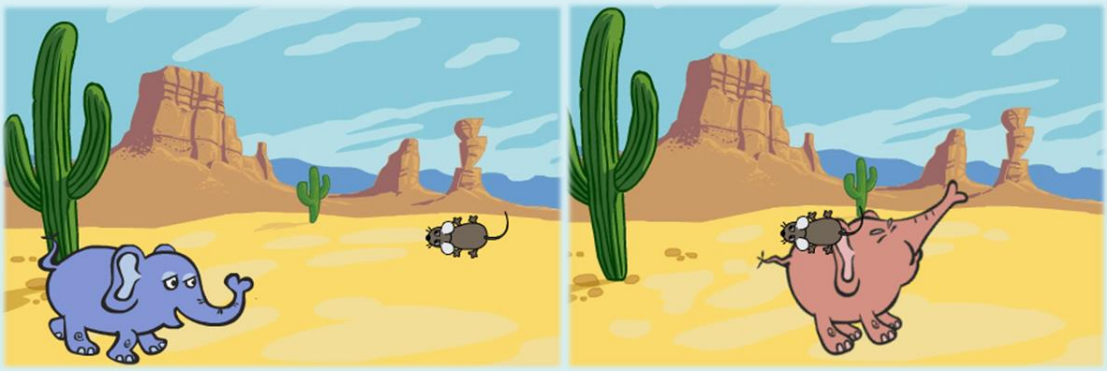
   Če se žoga dotakne golmana, ta reče, da je branil gol, in če se ga ne, reče, da mu ni uspelo. Kodo dodajamo v vratarjevo polje za kodo, saj bo ta preverjal, ali se ga žoga dotika. Žogo dodamo na prizorišče tako, da jo postavimo z miško na neko mesto.

Dodatne možnosti

Otroci naj poskušajo ugotoviti, kako bi ves čas preverjali, ali je žoga že v golu, ne le prvič, ko se žoga dotakne gola.

Rešitev

slon se prestraši



Naloga

Slon naj se ves čas premika v smeri proti miši. Miš lahko premikamo tako, da kliknemo nanjo z računalniško miško in jo premaknemo. Če se slon dotakne miši, naj potem zatrobi tako, da se slika slona zamenja s sliko slona, ki trobi.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati osnovne bloke in poiskati nove,
- osnove pogojnih stavkov.

Učenci v okviru naloge:

- uporabljajo zanke in pogojni stavek,
- nevede se naučijo osnov večnitnega izvajanja.

Materiali

http://scratch.mit.edu/projects/15107433/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava ozadja in izbira figure
   Pripravimo primerno ozadje, za figuri izberemo slona in miš.

3. Napišemo programsko kodo za slona

   Da se slon ves čas premika, je potrebno uporabiti zanko: “ponavljaj”. Za obračanje proti miši obstaja funkcija: “Obrni se proti x.” Da slon ob dotikanju miši spremeni izgled, je potrebno uporabiti pogojni stavek: “Če je x potem.” Kostum/izgled pa spreminjamo z ukazom: “Zamenjaj kostum na x.”

   Med menjanjem kostuma/izgleda je potrebno za nekaj časa počakati, da sploh opazimo, da se kostum menja, saj se drugače menja tako hitro, da sprememb ne moremo opaziti.

Dodatne možnosti

Slon naj po tem, ko potrobi, spet vrne rilec v prvotno stanje. Poleg tega lahko slon postane še rdeč od strahu ob dotiku z mišjo.

Rešitev

ODZIV NA DOGODKE

Kaj je to odziv na dogodke?

Če se udarim, me zaboli. Če se me dotakneš, to začutim. Odzivam se na dogodke. Tudi pri programiranju se figure odzivajo na podoben način. Lahko čakajo, dokler se jih ne dotakneš in potem nekaj izvedejo. Figura te lahko vpraša po imenu, številki ali kakšnem drugem podatku in nato z njim izvaja program. Predlagam ti, da rešiš nekaj nalog s tega področja in takoj ti bo jasno, kaj pomeni odzivanje na vnesene dogodke.

Trampolin

Naloga

Na trampolinu naj skače deklica, ki te vpraša, kako visoko naj skoči. Odgovori ji z vpisom številke in ona naj skoči ter se vrne nazaj na začetno mesto.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati osnovne bloke in poiskati nove,
- uporabljati koordinatni sistem.

Učenci v okviru naloge:

- spoznajo, kako se program odziva na dogodke, v tem primeru na vnos.

Materiali

http://scratch.mit.edu/projects/19827462/

Potek

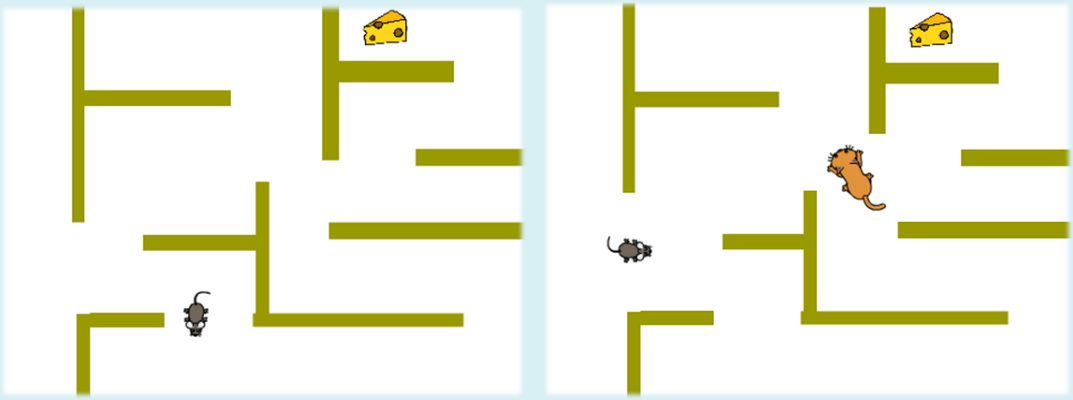
1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava ozadja in izbira figure
   Med figurami poiščemo trampolin in deklico, ki skače.
3. Uporaba blokov za zaznavanje
   Uporabimo blok iz kategorije zaznavanje, ki vpraša, koliko naj skoči v zrak. Nato naj s pomočjo spreminjanja y-koordinate skoči v zrak za toliko, kolikor uporabnik vpiše, in zatem skoči nazaj na začetno točko tako, da se y-koordinata zmanjša za vneseno število. Odgovor uporabnika dobimo v spremenljivki »odgovor«.

Dodatne možnosti

Nad deklico postavi še eno figuro npr.: sonce. Če se deklica dotakne sonca, naj reče: »Au, peče!«

Rešitev

Naloga – Labirint



Naloga

Pripravi labirint, vanj postavi miško, ki naj pride do sira. Takrat se igra zaključi.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti in narisati svoje ozadje ter izbrati figuro,
- uporabljati osnovne bloke in poiskati nove,
- osnove koordinatnega sistema in kote,
- uporabljati pogojne stavke,
- pisati programe za več elementov (večnitno izvajanje za dodatno nalogo).

Učenci v okviru naloge:

- utrdijo znanje o pogojnih stavkih,
- se naučijo pisati programe z odzivanjem na dogodke, kot je pritisk smernih tipk.

Materiali

http://scratch.mit.edu/projects/20133055/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava ozadja in izbira figure
   Ozadje pripravimo tako, da narišemo labirint. Poleg tega izberemo dve figuri, miš in sir, do katerega bo morala miš priti. Po potrebi ju zmanjšamo tako, da dvakrat kliknemo na ukaz:

»Nastavi na velikost 50%.«

3. Premikanje miši
   Miš naj se premika glede na pritisnjene tipke na tipkovnici. Uporabimo ukaz, ki bo nadaljnje ukaze izvedel ob dogodku: ko je pritisnjena tipka. Če bo pritisnjena tipka navzgor, se bo miška obrnila navzgor in šla naprej nekaj korakov. Enake ukaze napišemo za premik v desno, levo in dol.
4. Naj miš začne svojo pot na začetku labirinta
   Naj se miš na začetku prikaže in postavi na začetno točko zaslona. Tja jo postavimo z ukazom: pojdi na x in y.
5. Prepreči miški, da bi hodila čez stene labirinta
   Miši moramo preprečiti, da bi hodila skozi stene labirinta tako, da v kodo vstavimo pogoj, ki se ves čas izvaja. Če se dotakne stene labirinta (najlažje je, da vstavimo pogoj, če se dotika barve, kakršne so stene labirinta), naj gre na začetek zaslona. (Če se želijo v steno le zadeti in je ne želijo prečkati, naj se figura premakne za 10 korakov nazaj.)
6. Ustvari sir oziroma ciljno figuro
   Dodaj novo figuro, do katere bo morala miš priti, da bo dosegla zmago.
7. Ob zmagi spremeni ozadje zaslona in skrij vse figure
   Igralec zmaga takrat, ko se miš dotakne sira. To pomeni, da napišemo pogoj, ki se tudi mora ves čas preverjati, da se ob dotiku s sirom zamenja ozadje. Ne smemo pozabiti, jih potem prikazati ob pritisku na zeleno zastavico

Dodatne možnosti

Naj mačka med miškinim potovanjem lovi miš. Če se dotakne miši, naj se igra konča in ozadje zamenja. Mačko naj otroci poskusijo sprogramirati sami na podoben način, kot smo napisali ukaze za miš. Mačka se bo morala premikati ves čas in ne ob ukazih s tipkovnice.

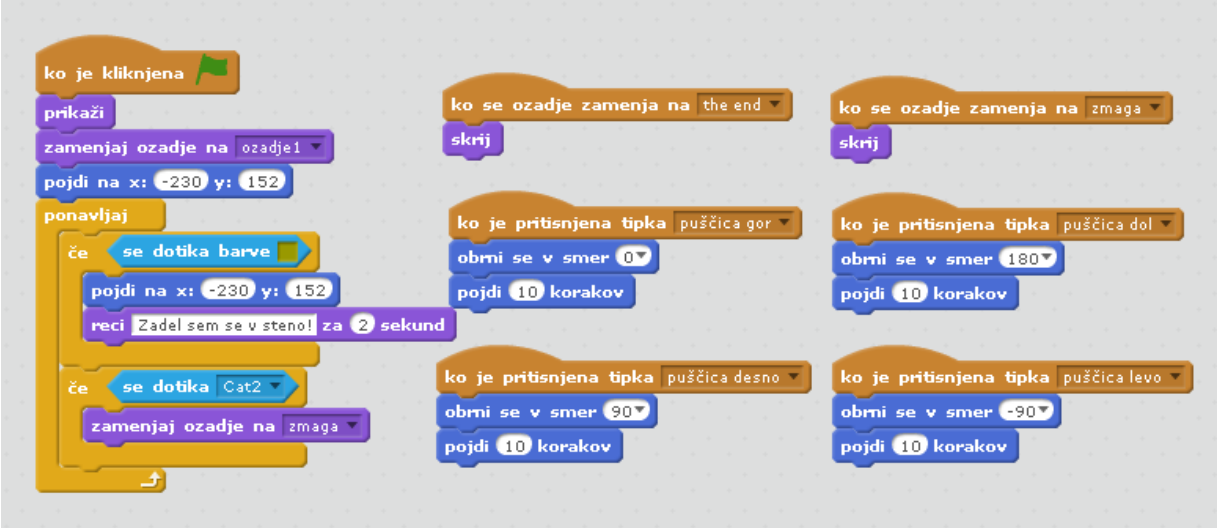
Rešitev

Koda za sir:

Koda za miš:

Dodatno

Koda za miš:



Koda za sir:

Koda za mačko:

SPREMENLJIVKE

Kaj so spremenljivke?

Spremenljivka je navadno neka vrednost, ki se spreminja.
To je na primer število košev, ki jih zadane košarkaš, in se poveča vsakič, ko košarkaš zadane koš. To je lahko tudi rezultat v igrici.
Spremenljivke uporabljamo, ko želimo na začetku določiti neko vrednost in jo nato uporabljati.
Na primer, če seštevamo števili a + 4, lahko rečemo, da je spremenljivka a število 2. To pomeni, da je a + 4 = 6. Če spremenimo vrednost a, bo rezultat drugačen. a je tako v tem primeru spremenljivka.

Naloga – klikni ptiča

Naloga

Ptič naj leti po površini in se odbija, če prileti do roba. Vedno, ko kliknemo zastavico, naj se spremenljivka z imenom Ptič nastavi na 0. Če pa z miško kliknemo na figuro, naj se vsakič, ko kliknemo ptiča, prišteje spremenljivki Ptič ena točka.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,

- uporabljati bloke in poiskati nove
- uporabljati zanke, pogojne stavke,
- sprogramirati zaznavanje dotika z miško in se odzvati na dogodek.

Učenci se v okviru naloge:

- naučijo uporabo spremenljivk.

Materiali

http://scratch.mit.edu/projects/19841787/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure
   Pripravimo ustrezno ozadje in primerno figuro.

3. Naj se ptič ves čas premika in odbija od roba
   Ptič naj se v neskončni zanki premika in odbija od roba. Na začetku ga je potrebno malo zasukati, da se ne odbija le levo in desno.

4. Dodaj spremenljivko
   Spremenljivko ustvarimo na področju s Podatki. Tam ustvarimo novo spremenljivko, ki jo poimenujemo. Ko kliknemo zastavico, naj se ta spremenljivka nastavi na 0, vsakič ko kliknemo figuro, pa naj se spremeni za 1.

Dodatne možnosti

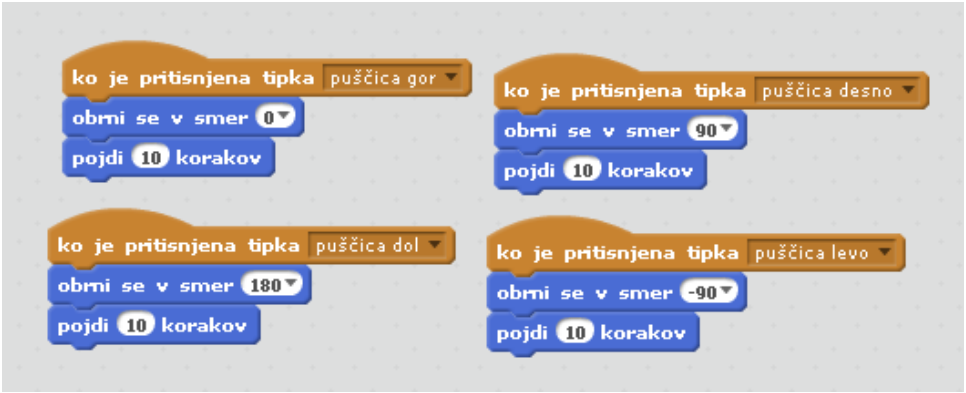
Ustvari še eno figuro, na primer hrošča, ki se bo moral umikati ptiču s pomočjo tipk levo, desno, gor in dol, da ga ptič ne bo pojedel. Ptič naj še vedno leta tako kot prej. Vendar naj ima igralec sedaj na začetku 20 točk in ob vsakem dotiku hrošča s ptičem, naj izgubi eno točko. Ko ima nič točk, naj se igra ustavi.

Rešitev

Dodatno

Rešitev za ptiča:

Rešitev za hrošča:



Naloga – Kaj je večje?

Naloga

Ustvari dve spremenljivki, ki ju lahko z drsniki spreminjaš. Maček naj pove, katero število je večje oziroma enako. Osnovni del naloge naj bo, da Maček pove, če je prvo število večje, če le-to ni, naj reče: »Drugo število je večje ali enako!«

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- znajo uporabljati osnovne bloke in poiskati nove,
- delati z operacijami večje, manjše ali je enako.

Učenci v okviru naloge:

- utrjujejo uporabo pogojnih stavkov in spremenljivk.

Materiali

http://scratch.mit.edu/projects/19841052/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

2. Priprava ozadja in izbira figure

Po želji otrok.

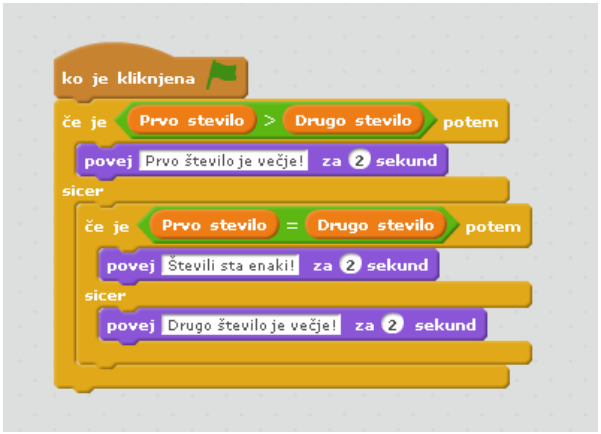
3. Ustvarimo dve spremenljivki
   Spremenljivke ustvarimo v kategoriji blokov z imenom Spremenljivke. Tam ustvarimo dve novi spremenljivki in ju po svoje poimenujemo. Ob tem obkljukamo kvadratek ob imenu spremenljivke, da jo naredimo vidno na prikaznem polju, tudi ko se program izvaja. Če dvakrat kliknemo na spremenljivko na prikaznem polju, se pojavi drsnik in tako lahko izbiramo vrednost spremenljivke.

4. Uporaba pogojnega stavka (Če potem , … sicer)
   Najprej se je potrebno pogovoriti, kako bo maček odgovarjal. Če bo prvo število večje od drugega, bo rekel: »Prvo število je večje od drugega.« Če pa bo drugo število večje ali enako, bo rekel: »Drugo število je večje ali enako.« Naj otroci najprej sami poskušajo najti pravi blok in sami vstaviti primerne ukaze.

Dodatne možnosti

V dodatnem delu naj maček prepozna, katero število je večje, manjše ali pa sta števili enaki.

Rešitev



VEČNITNO IZVAJANJE

Najboljši primer večnitnega izvajanja so mame. Mame lahko naenkrat telefonirajo, pomivajo posodo in hranijo otročička. To pomeni, da naenkrat izvajajo več programov.

Vendar večnitno izvajanje ni le to, da mama nekaj dela naenkrat. Večnitnost je tudi to, da ob istem času zvoni telefon, zunaj laja kuža in se otrok igra na igrišču.

V nalogi labirint si že uporabil večnitno izvajanje, brez da bi to vedel, tako da veš, da ta snov ni tako zahtevna.

Na poti v šolo

Naloga

Fant, ki hodi po ulici, naj se glede na pritisnjene tipke (puščica levo, desno, gor, dol) premika v smeri puščic. Poleg tega lahko menja tudi kostum, na primer, ko gre levo: naj bo obrnjen v levo, v desno pa ravno obratno. Za premikanje gor in dol naj bo postavljen pokonci. Če se medtem ko hodi, dotakne mačke, ta zamijavka, če pa se dotakne psa, le-ta zalaja.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati bloke in poiskati nove,
- pripraviti več figur,
- uporabljati pogojne stavke,
- uporabljati neskončno zanko,
- uporabljati koordinatni sistem v Scratchu.

Učenci v okviru naloge:

- ponovijo uporabo pogojnih stavkov,
- se spoznajo z večnitnim izvajanjem, vnosom ukazov preko tipkovnice, odzivanjem na dogodke,
- uporabljajo zvočne učinke v Scratchu,
- naučijo se spreminjati videz figuram med izvajanjem programa.

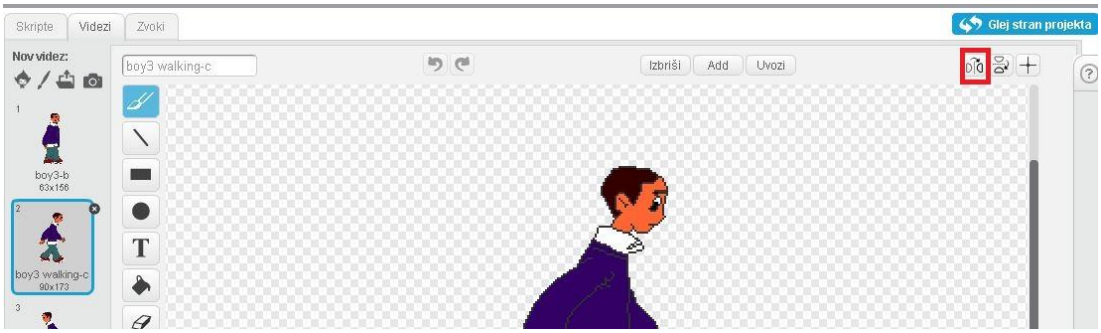
Materiali

http://scratch.mit.edu/projects/15109442/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava ozadja in izbira figure
   Pripravimo primerno ozadje, za glavno figuro izberemo fanta, dodamo še psa in mačko.
3. Figura fanta naj se premika glede na pritisnjene tipke
   Glede na pritisnjene tipke naj fant hodi gor, dol, levo, desno. To znamo sprogramirati že iz prejšnjih nalog.
4. Fant naj se obrača glede na smer hoje
   Fant se ne bo obračal tako kot drugod, ampak bo ob premikanju gor in dol ostal obrnjen proti nam, spremenila se mu bo le y-koordinata, pri hoji v levo in desno pa mu zamenjajte videz na dečka, ki hodi v levo ali desno – primerno smeri, v katero gre. Pomagajte si z ukazom zamenjaj videz. Primerne videze poiščete v rubriki z videzi in jih dodate drugega za drugim v eno figuro na ta način:

Z ene strani na drugo ga lahko preslikate v urejevalniku slik z ukazom:



5. <u>Če se fant dotakne živali, naj se ta oglasi</u>
   Pes in mačka naj ves čas v zanki čakata, kdaj se ju bo dotaknil fant. Če se ju dotakne, naj pes zalaja, mačka pa naj zamijavka.

Dodatne možnosti

Ko se deček povzpne po stopnicah, se na vrhu stopnic njegova podoba zmanjša na 50%.

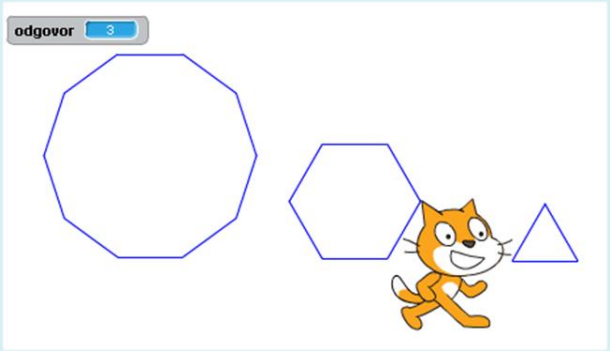
Rešitev

Rešitev za psa in mačko:

Rešitev dodatne naloge:

SESTAVLJENE NALOGE

Večkotniki



Naloga

Maček naj uporabnika najprej vpraša, koliko kotov naj ima lik, ki ga bo narisal. Večkotnike riše tako, da spusti svinčnik in nariše zahtevani večkotnik.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati bloke in poiskati nove,
- uporabljati bloke za zaznavanje: vprašanja in odgovore,
- uporabljati zanke,
- razumeti kote v likih, če ne, jim mora mentor kote razložiti.

Učenci v okviru naloge spoznajo:

- računanje kotov v večkotnikih,
- uporabo zanke s spremenljivko,
- uporabo spremenljivke, ki jo prejme iz odgovora.

Materiali

http://scratch.mit.edu/projects/19840667/

Potek

1. Pokažemo končni izdelek

Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.

3. Narišemo kvadrat

Maček naj najprej nariše kvadrat. Riše tako, da spusti svinčnik in se nato premika po pravi poti. Kvadrat narišemo tako, da se premaknemo za nekaj korakov naprej in obrnemo za 90°. To vse skupaj ponovimo štirikrat. Risanje kvadrata lahko razložimo kar s konkretnim primerom, tako da eden izmed učencev hodi in se premika, kakor mu ukažejo sošolci.

3. Ugotovimo formulo za izračun velikosti kotov mnogokotnika

Če vemo, da je vsota vseh notranjih kotov mnogokotnika 360°, potem lahko z nekaj sklepanja ugotovimo tudi formulo za ostale mnogokotnike. Otroci naj sami ugotovijo, da je velikost kota 360°/število stranic, stranice pa narišemo tolikokrat, kolikor stranični je lik.

4. Dodamo še vprašanje: »Koliko kotov naj ima lik?«

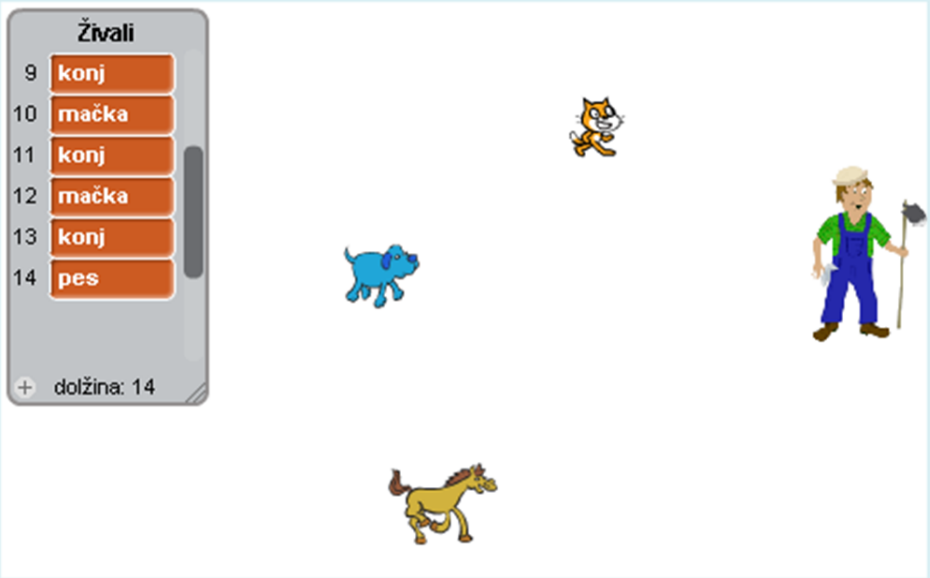
Vprašanje podamo v bloku za zaznavanje, namesto števila kotov, ki smo jih do sedaj vpisovali na roke, pa sedaj vstavimo spremenljivko odgovor.

Dodatne možnosti

Kako bi narisali čim bolj krogu podoben lik? Poizkusite, koliko stranični naj bo lik.

Rešitev

Kmet lovi živali



Naloga

Izdelaj igro, kjer kmet lovi svoje živali. Mucke, psi in konji tečejo z leve strani ekrana proti desni, kmet pa jih mora uloviti tako, da se jih dotakne. Ko se jih dotakne, izginejo in na seznamu živali se pojavi, kaj je ujel.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati zanke, pogojne stavke, pisati kodo za več figur,
- uporabljati koordinatni sistem,
- uporabljati ukaz z naključnim številom na nekem intervalu.

Učenci se v okviru naloge:

- naučijo uporabe tabel.

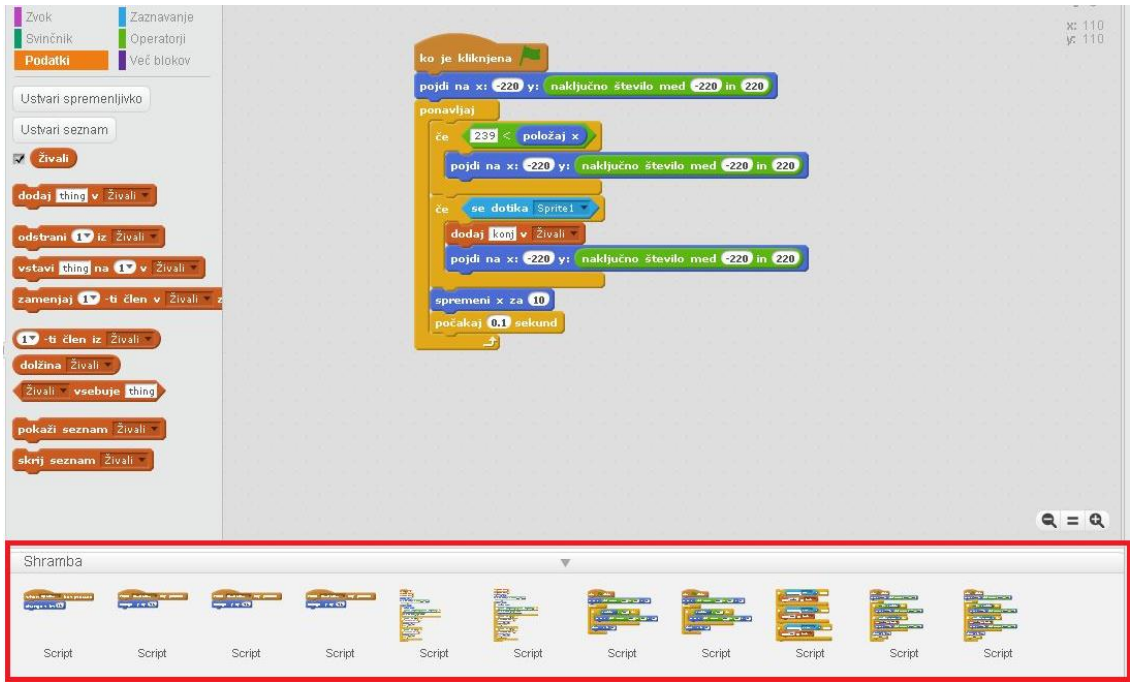
Materiali

http://scratch.mit.edu/projects/24976200/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava figur
   Pripravimo glavno figuro – kmeta ter vsaj tri različne živali.
3. Ustvarjanje seznama
   Seznam ustvarimo pri »Podatkih« in sicer kliknemo: »Ustvari seznam,« ter tam vpišemo ime seznama, ki je v našem primeru: Živali. V seznam bomo kasneje vpisovali, katere živali je kmet že ujel.
4. Živali naj se premikajo po zaslonu od leve proti desni in ko so na robu, naj se ponovno naključno pojavijo
   Živali naj se naključno pojavijo na levem robu zaslona. To naredimo z ukazom pojdi na določen x in naključni y. X je določen zato, ker vemo, kje je rob ekrana, y pa naj bo naključen od zgornjega roba do spodnjega. V neskončni zanki naj se premikajo figure naprej, poleg tega moramo v neskončni zanki preverjati tudi, če je žival že na robu ali če se figura dotika kmeta. Če je figura že na robu, jo spet naključno vrnemo na rob zaslona in naključni y, če pa se dotakne kmeta, naj se doda spremenljivka z imenom živali v seznam Živali ter šele nato nadaljuje na levi rob zaslona. Kodo lahko potem shranimo v shrambo in jo skopiramo še v druge živali, da nam je ne bo potrebno večkrat pisati. Spremeniti bo potrebno le ime živali, ki jo dodajamo v seznam.

   Shrambo najdemo tu:



5. Kmet naj se premika glede na vnos s tipkovnice gor in dol
   Če kliknemo tipko gor, naj se kmet premakne navzgor po ekranu, in če dol, naj se premakne navzdol. Levo in desno naj se ne premika.
6. Ob začetku nove igre naj se seznam v tabeli izbriše
   Ob kliku na zastavico pobriši vse elemente v tabeli.

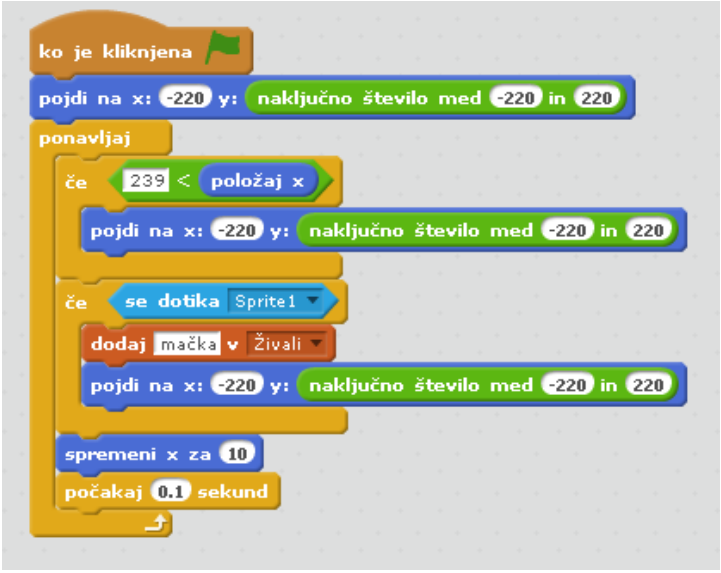
Dodatne možnosti

Naj se igra konča, ko prvi dobi 21 točk, in naj računalnik razglasi zmagovalca.

Rešitev

Rešitev za kmeta:

Rešitev za živali:



Ping pong

Naloga

Izdelaj igro ping pong/namiznega tenisa. Na levi in desni postavi loparja, robova igrišča pa pobarvaj vsakega z drugačno barvo, da boš lahko štel točke. Žoga naj se odbija od robov in loparjev, če pa pade na obarvani rob, naj se nasprotniku prišteje točka, žoga pa naj servis začne s srede igrišča v naključno smer.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,
- uporabljati zanke, pogojne stavke, pisati kodo za več figur.

Učenci v okviru naloge spoznajo uporabo:

- aritmetičnih operacij,
- naključnih števil,
- logičnih operacij.

Materiali

http://scratch.mit.edu/projects/15108592/

Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava ozadja in izbira figure
   Ozadje pripravimo tako, da pobarvamo eno stran roba s svojo barvo, drugo stran pa z drugo. Tako bomo namreč lahko šteli točke, ko bo žoga padla izven polja. Za figuro vzamemo žogo, ki jo bomo odbijali. Poleg tega narišemo še dva loparja.
3. Napišemo program za žogo
   Vedno naj začne izvajanje na sredini in se obrne v naključno smer. Nato naj se ves čas premika naprej in odbija, če je na robu.
4. Ob dotiku loparja, se odbij
   Če se žoga dotakne loparja, enega ali drugega, se mora odbiti. Napiši pogojni stavek tako, da vmes uporabiš ukaz ali (če se dotakne enega loparja ali drugega). V pogoju zapiši, da se žoga odbije v nasprotno smer, od koder je prišla.
5. Dodaj štetje točk
   Ob dotiku zunanje črte prištej točko drugemu igralcu. Poleg tega naj gre žoga na sredino in od tam začne padati v naključno smer.
6. Naj se loparja premikata glede na pritisnjene tipke
   Loparja naj se premikata le dol in navzgor. Za vsakega določite tipke, ki ju bodo premikale in to tudi sprogramirajte.

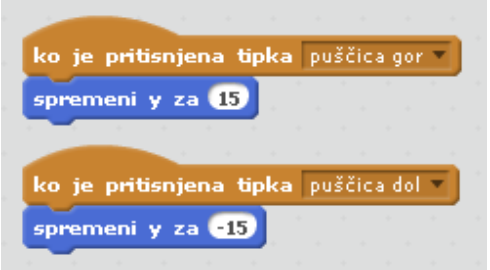
Dodatne možnosti

Naj se igra konča, ko prvi dobi 21 točk, in naj računalnik razglasi zmagovalca.

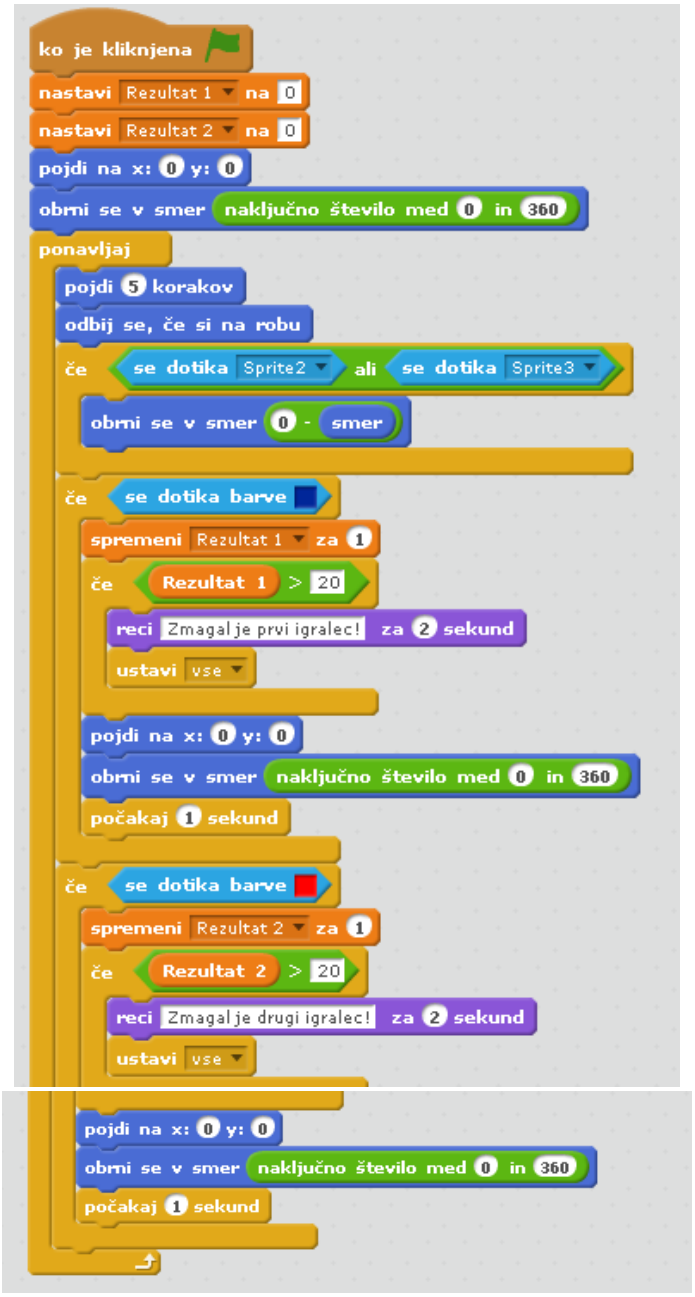
Rešitev

Koda za žogo:

Koda za lopar (za še en lopar je potrebno izbrati le drugačne tipke) :



Dodatno
Koda za žogo:

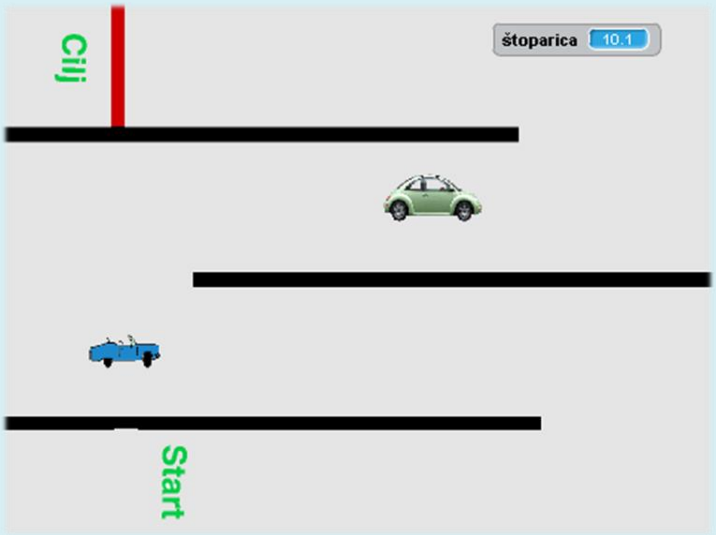


SPOROČILA

Kaj so sporočila?

Sporočilo je nekaj, kar si dva osebka sporočita med seboj. To pomeni, da eden enemu nekaj govori, drugi ga posluša in prejme njegovo sporočilo.
Seveda se lahko nato njuna vloga zamenja. Tisti, ki je sporočilo prejel, lahko odgovori in to naredi tako, da pošlje sporočilo, drugi na sporočilo čaka in ga prejme.

Dirka v dvoje



Naloga

Dva avtomobila tekmujeta, kdo bo prej prišel do cilja. Ko štoparica pokaže naključen čas od 0 do 5 sekund, naj avtomobila začneta voziti naravnost, igralca pa naj ju usmerjata s smernimi tipkami levo in desno.

Predznanje in cilji

Predpostavljamo, da učenci že znajo:

- pripraviti ozadje in izbrati figuro,

- uporabljati zanke, pogojne stavke, pisati kodo za več figur,
- narediti igro za več igralcev.

Učenci v okviru naloge spoznajo uporabo:

- časovne usklajenosti in sinhronizacije,
- pošiljanja sporočil.

Materiali

http://scratch.mit.edu/projects/20906732/

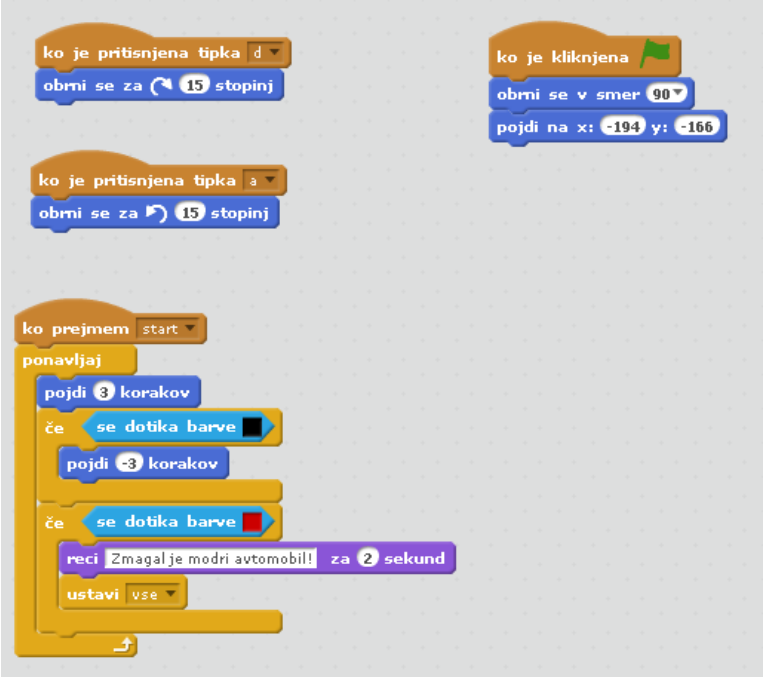
Potek

1. Pokažemo končni izdelek
   Na povezavi z materiali dostopamo do programa in pokažemo, kako program deluje, vendar programske kode ne pokažemo.
2. Priprava ozadja in izbira figure
   Ozadje pripravimo tako, da narišemo dirkalno stezo, cilj pa označimo z rdečo črto.
3. Napišemo program za avtomobila
   Avtomobila se bosta ves čas premikala naprej, s smernimi tipkami ju bomo sukali le levo in desno. Tako je potrebno napisati le ukaze za zasuk v levo in desno. Ob kliku na zastavico naj se avtomobila postavita na start in obrneta v pravilno smer.
4. Nastavi štoparico in sporoči, ko bo minilo 5 sekund
   Program lahko napišemo kar pri skripti za ozadje. Počaka naj, dokler čas na štoparici ni večji od naključno izbranega števila med 0 in 5. Za čas na štoparici uporabite spremenljivko, ki je shranjena v sklopu zaznavanje. Po pretečenem času na štoparici naj program objavi, da se je dirka začela.
5. Po prejeti objavi, da se je start začel, začni s premikanjem
   Ko avtomobila prejmeta obvestilo, da se je dirka začela, se začneta premikati naprej. Če se dotakneta roba cestišča, ju je treba pomakniti toliko korakov nazaj, kolikor se premikata naprej v enem ukazu. Če pa se dotakneta rdeče barve, je tisti, ki se je dotakne prvi, zmagovalec. Takrat avtomobil sporoči, kateri avto je zmagal in program se ustavi.

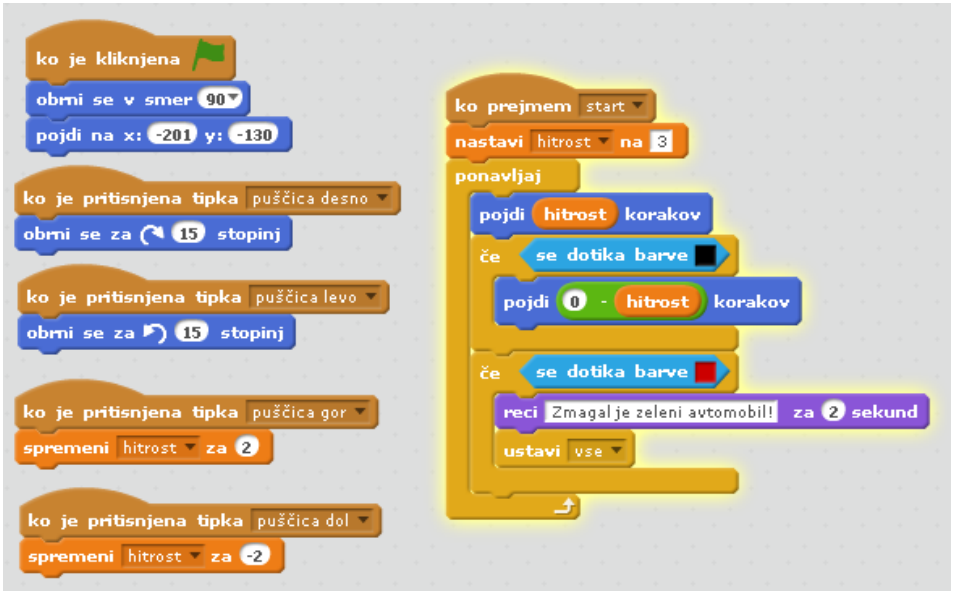
Dodatne možnosti

Povečuj hitrost s pritiskom na gumb.

Rešitev



Dodatno



Dodatne naloge

Med krožkom predlagam, da kdaj uporabite tudi igre, ki jih predlaga program Vidra in spodbuja računalništvo brez računalnika. Igre lahko izvedete ob začetku ure ali za sprostitev, če programirate dlje časa.
Dobra naloga je tudi, da se otroci presedejo za dve mesti in zamenjajo računalnike ter nadaljujejo s programiranjem sošolčeve kode. S tem se naučijo brati kodo drugih, trudijo se razumeti, kaj koda pomeni, obenem pa tudi oni začnejo pisati preglednejšo in bolj urejeno kodo.
Predlagam tudi, da kdaj vmes pokažete kakšno nedelujočo kodo in morajo udeleženci ugotoviti, kaj je v njej narobe. Tudi ta naloga spodbuja razumevanje kode drugega in spodbuja iskanje napak v kodi, kar bo otrokom prišlo prav pri pravem programiranju.

Zaključek

Priročnik je nastal kot diplomsko delo. Komentarje in odzive na naloge lahko posredujete na: petra.mihalic92@gmail.com.

Viri

- Openclipart - Farmer. Dostopno na: http://openclipart.org/detail/46291/farmer-by-j_alves